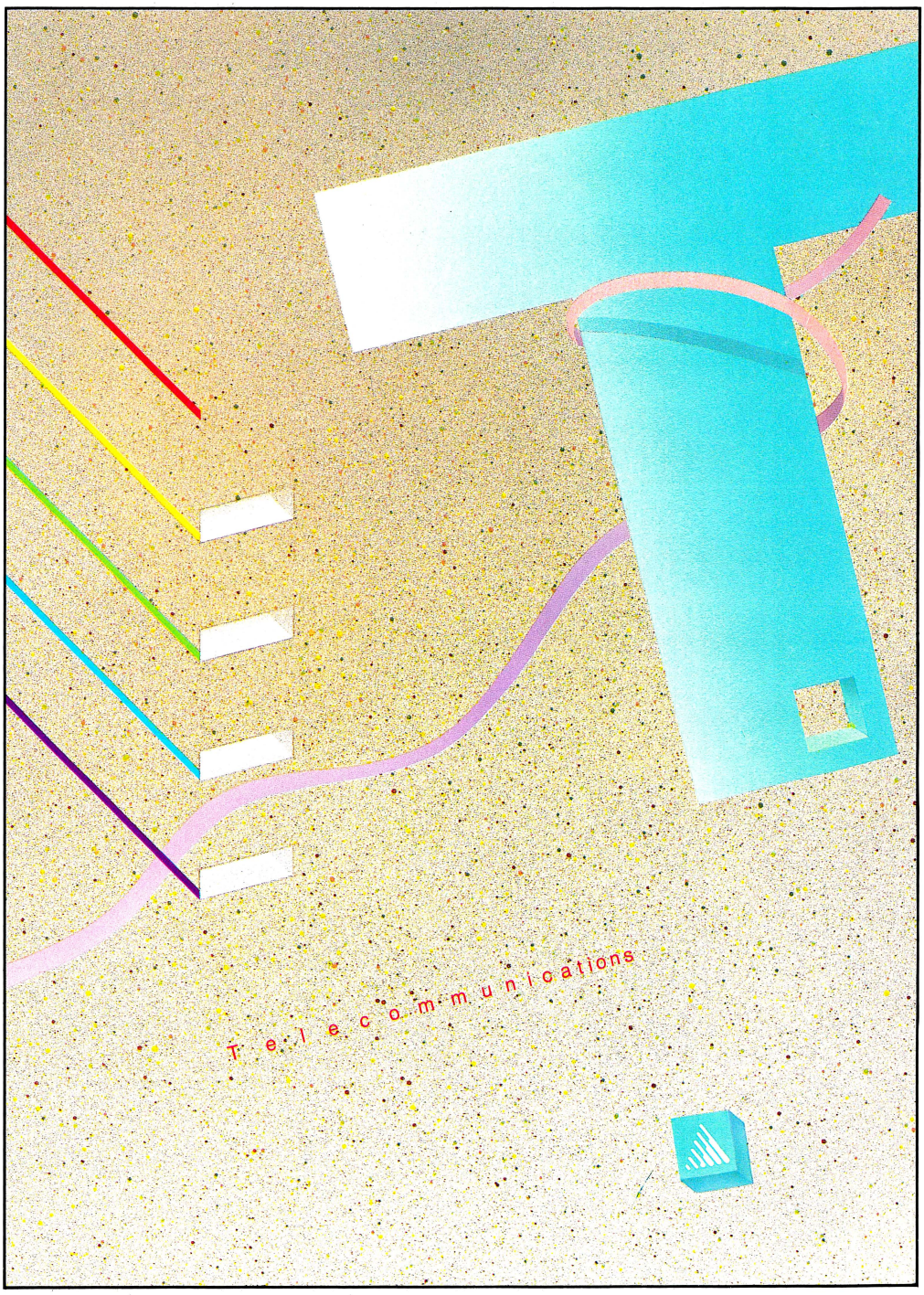
AN ASHTON-TATE SOFTWARE SUPPORT PUBLICATION

APR 21 1986

APRIL 1986

TechNotes

- Tele-communications in Framework II
- dBASE III PLUS
  - Catalog Management
  - Vertical Form LIST
- dBASE TOOLS FOR C
- MultiMate Technical Support Information



TechNotes

TABLE OF CONTENTS

FEATURES

3 Telecommunications Primer
by Kent Irwin

9 Arrays in dBASE TOOLS FOR C
by Robert Boies

20 Operating dBASE III PLUS in a 256K Environment
by Gail Kulikowsky

22 LISTing a Database File in a Vertical Form
by Stanley Ballanger

26 An Example of Catalog Management
by Quinn Wildman

34 A Guided Tour of the Library Cabinet
by Paul Bukauskas

39 How to Use Boxes in Word Frames
by Gail Kulikowsky

DEPARTMENTS

1 Read Me First
by Laurie Miller

19 Corrigenda

42 Usage Tips

47 Anomalies

50 MultiMate Technical Support Information

55 Letters

59 Update Policies

66 Index

READ ME FIRST

by Laurie Miller

In this issue of ***TechNotes***, we are featuring Framework II Telecommunications. Kent Irwin's article, "Telecommunications Primer," the first in a series, familiarizes you with telecommunicating in the Framework environment. This month's article covers hardware setup—understanding the modem's switches, properly setting them, and hooking up the modem to the computer. Kent also steps us through software setup via the Framework II Apps: Telecommunications: Setup and Setup Options menus. I highly recommend this series to all Framework II users, beginning or advanced.

For our dBASE III PLUS readers, this issue offers you Quinn Wildman's article, "Catalog Management," and Stanley Ballenger's article, "LISTing a Database File in Vertical Form." "Catalog Management" discusses the new dBASE III PLUS Catalog feature and how you can ues it to manipulate your database files more efficiently. Stanley presents a program that demonstrates dBASE III PLUS's new FIELDS() and DBF() functions. The program, Listform.PRG, customizes a database file LISTing.

If you're a C programmer, or are interested in learning C, read Robert Boies's "Arrays in dBASE TOOLS." This article, the first in a series on interfacing dBASE III PLUS with C, is based on a new Ashton-Tate product called *dBASE TOOLS FOR C: The Programmer's Library*. In this first installment, Robert provides a detailed discussion of arrays—what they are, how they are used, and how to create them in dBASE III PLUS. This is another highly recommended article for novices and experts alike.

It's official! Ashton-Tate and MultiMate International are now one. As such, this is the first issue of ***TechNotes*** to contain technical support information on MultiMate products! Included is information on how and when to contact MultiMate technical support, other resources you can use to get answers to your technical questions, and information about MultiMate's newest version of Advantage which offers column features and a thesaurus. We hope you find this introduction of MultiMate technical information useful. Please continue to contact MultiMate for technical support on MultiMate products and Ashton-Tate for dBASE and Framework.

Enjoy this issue! And remember to fill out and return to us the Reader Survey, located toward the back. We always appreciate your comments and suggestions.

Publisher
Ashton-Tate Software Support Center
Jay Hanson, Director
Roy Moore, Manager

Editor-in-Chief
Laurie Miller

Editors
Kent Irwin, Christopher White

Assistant Editor
Nicole Hannes

Technical Staff
Stanley Ballenger, Jeffrey Bersin, Olivier Biggerstaff, Robert Boies, Paul Bukauskas, Richard Bunter, Clay Carley, Ray Cortenbach, Fred Crownover, Michael Fiore, Richard Goepel, Peter Gold, Stephen Goodman, Doug Haman, Mike Hedblom, Jack Hirsch, David Howlett, Errol Jackson, Matt Johnson, Rosaline Keenan, Gail Kulikowsky, Steven Kurasch, Chris Lovasco, Jerry Mendes, Elizabeth Miranda, Joseph Perret, John Regen, Karen Robinson, Steve Silverwood, Lisa Wanek, Quinn Wildman, Tom Williams, Leonard Zerman

Administrative Staff
Carol Jackson, April Larson, Kathy Smith

Art Staff
D. A. Gray Bare, Darrell Evers, Stephen Goodman, Dusty Lee

Typesetting
Carol Heys, Renée Maynard

Index
Nicole Hannes

Product Development
Forefront Development Center:
Marilyn McMahon, Greg Stikeleather
Glendale Development Center:
Debby Moody

DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under Anomalies) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Ave.
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes***.

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate office for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind Ashton-Tate.

TechNotes (ISSN 0883-6493) is an Ashton-Tate Software Support Center publication containing technical reference information about three of Ashton-Tate's products: dBASE II®, dBASE III PLUS®, and Framework II™.

Entire contents Copyright © Ashton-Tate 1986. All rights reserved. Commercial reproduction of the material appearing in ***TechNotes*** is forbidden without prior written permission. Direct all requests for reproduction permission to Ashton-Tate Software Support Center, 20101 Hamilton Ave., Torrance, CA 90502-1319 ATTN: Laurie Miller.

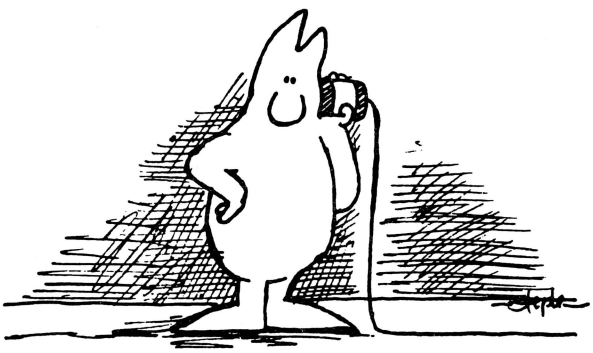
TechNotes is published monthly at a subscription price of $50.00 for one year. Additional rates: Canada and Mexico, $65.00; other countries, $75.00. Monthly issues are available in back issues for $10.00 each (1985 issues) and $7.00 each (1986 issues). ***TechNotes*** is published by the Ashton-Tate Software Support Center, 20101 Hamilton Ave., Torrance, CA 90502-1319. POSTMASTER: Send subscription inquiries and address changes to ***TechNotes*** Subscription Service, P.O. Box 3729, Escondido, CA 92025-0929.

dBASE II and dBASE III are registered trademarks of Ashton-Tate. dBASE, RunTime, FRED, Framework, and ASHTON-TATE are trademarks of Ashton-Tate. Some of the trademarks of other companies mentioned in this issue appear for identification purposes only.

SOURCE AT6

Telecommunications Primer

by Kent Irwin



One of the most exciting new features of Framework II is the Telecommunications module on the new Apps pull-down menu. This powerful module brings the wealth of electronic information networks right onto your desktop. Up-to-date news and financial information, reference material, and literally thousands of public domain programs are at your fingertips. The integration of this public information into your Framework II documents is now a smooth and natural process because you never have to leave Framework II to get it! I hope to encourage your interest in this rapidly expanding electronic medium by helping you get started with the basics.

Hardware Setup

To use telecommunications, you must have properly configured hardware. In addition to your computer you need a modem, a way to connect the modem to your computer, and a working phone line. This is essentially a simple process, but many people have difficulty with it because of the many different products offered in today's market. During this setup process you may find it useful to fill out the hardware summary on pages GS-7 and GS-8 of *Getting Started Framework II*. This will help you keep track of your current configuration and make it easier for you to change things later.

Begin by ensuring that your modem is properly set up to work with your computer. Most modems have switches or jumpers which "customize" the modem to both the computer and your phone lines. Look on page 2-5 of *Advanced Topics Framework II* to find the switch settings for your modem. If your modem isn't listed, this article may help you find the correct settings. It is important that you have the modem manual handy during this stage, because that is really the only place you will find a good list of what each switch or jumper will do. The Hayes Smartmodem 1200 has become a standard for modems, and so will be our example here. The Smartmodem has eight numbered switches which are accessed by removing the front cover (the cover over the status lights). Generally, switches 1, 2, 4, 5, 6, and 7 should be on (up) and switches 3 and 8 should be off (down) for the Smartmodem to work correctly with Framework II. These switch numbers are not the same on all modems, so I will discuss the function of each, and possible problems they can cause Framework II. If you are not using a Smartmodem, be sure to reference your modem's manual for the relevant switches discussed here.

Switch 1 on the Smartmodem is very important. It controls the operation of the modem control signal, called Data Terminal Ready (DTR). The position of switch 1 tells the Smartmodem either to ignore the DTR signal from the computer, or to allow the DTR signal to control the modem. Since Framework II uses DTR to turn the modem on and off, it is important that the switch be set to the position that allows DTR to control the modem. If your modem manual uses a phrase like "forced on" with regard to the default setting of this switch, then set it the other way. On the Smartmodem, this means switch 1 is up. If it is set incorrectly, you will notice that Framework II can never hang up the phone.

Switch 2 tells the Smartmodem to talk to the computer in either words or numbers. Since you want to know what the modem is saying when there is a problem, you will probably want this switch to be up (word result codes).

Switch 3 enables or disables the result codes referred to for switch 2. Framework II will work best with word result codes enabled, so put this switch down.

Switches 4 and 5 are not terribly important for the beginner. The Smartmodem gets its name from the fact that it has its own command set, and you can actually use Framework II to talk directly to the modem. If you want to do this, set switch 4 up. If you intend to call in to Framework II from another computer, your modem has to be able to answer the phone. If you set switch 5 up, the Smartmodem will answer the phone after a certain number of rings.

Switch 6, like switch 1, is very important: it controls the "carrier detect" ability of the modem. A carrier is the high-pitched scratchy sound that two computers use to communicate. If you pick up the phone during a telecommunications session, you will hear this sound. After your modem has dialed a phone number, it "listens" for this carrier, and when it is identified, the two computers begin to communicate. Switch 6 determines whether the modem will "listen" for a carrier, or just assume that the carrier is there. Framework II does need to "listen" for a real carrier, so you want switch 6 to be set up. Again, choose the switch setting that your manual describes as NOT forcing carrier detect on.

Switch 7 tells the Smartmodem what kind of phone line it is connected to, either a single- or multiple-line phone jack. A single-line phone is like most home phones or dedicated office phones. A multiline phone can handle several calls at once, like many office phones that have blinking buttons and generally a hold feature. For single-line phones, set switch 7 up, for multiline, set it down.

Switch 8 will enable or disable the "smart" commands of the Smartmodem. Generally, you will want to enable the commands, so set this switch down.

There are many ways these settings will differ on modems other than the Smartmodem 1200. For instance, the Hayes 1200B internal modem has two different models. One has only three switches and the DTR option is not a switch but a jumper; the other has six switches and the DTR is controlled by switch 4. The Hayes Smartmodem 2400 has no physical switches at all! However, the above discussion still applies in principle, and specific information about your modem can be found in the manufacturer's documentation. A partial list of settings for many common modems can be found in *Advanced Topics Framework II* on page 2-5. The steps to setup a Hayes Smartmodem 2400 are found on page 2-6.

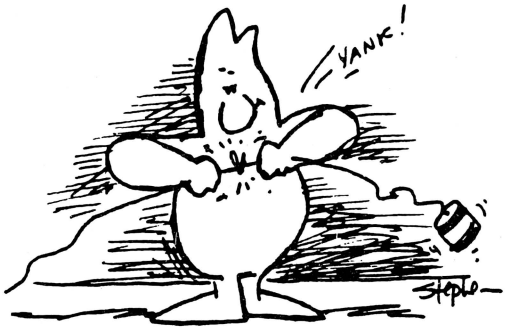
Now that the modem itself has been configured, you must connect it to your computer. A modem is considered a serial peripheral device, which means that it must be connected to the computer through a "port". Serial ports in PC/MS-DOS are labeled COM1: or COM2: and many machines are constructed with a COM1: connector on the back panel. If you have an internal "board"-type modem, you have an easy task. These devices have a jumper or switch on them that forces them to use either the COM1: or COM2: port. Once you have made that selection, simply plug the card into an empty expansion slot in your computer. The important thing to remember is that you must choose a serial port for your modem that is not already used by some other device connected to your computer. Sometimes it is necessary to change or disable the serial port that comes attached to your multifunction board (such as AST Six-Pack) in order to avoid conflicts between serial ports.

External modems are connected to the computer with a serial cable. If you already use a serial printer connected to COM1:, you will have to add a circuit board to your machine to give you a COM2: port. Many memory expansion boards will also give you an additional serial port, so examine closely the hardware you have before buying more. The cable you will need is often called a "modem" cable because of its most common application. It consists of 25 wires with a male connector on one end and a female connector on the other. Many computer stores will sell you a ready-to-go cable that will spare you the tedium of constructing one. However, if you feel adventurous and have the tools handy, you can examine the cable diagram on page 2-2 of *Advanced Topics Framework II* and try making your own. Only try this if you're experienced with connectors. Only eight of the 25 wires are actually used. Pins 2 through 8, and 20, on one connector run directly through to the corresponding pins of the other connector. Once you have an appropriate cable in hand, simply connect one end to the computer's serial port, and the other to the modem.

The final hardware adjustment is the simplest. Plug your modular phone jack into the modem. Some modems have two such jacks on their back panel; one is connected to the phone cable leading into the wall and the other is for the cable going to your phone. The latter is only needed if your computer is sharing your voice line or if you would like Framework II to dial your voice calls for you.

We can now safely proceed to using Telecommunications in Framework II.

Software Setup



Perhaps the easiest task you can perform with your modem and Framework II is **Point & Dial Voice Call**, found in the Apps:Telecommunications submenu. If you have a handset or speaker-phone connected to the same line as your modem, you can use this command to dial the phone for you. Once the number has been dialed, Framework II will prompt you to pick up the phone and press the **Spacebar**. Pressing the **Spacebar** causes Framework II to hang up the modem, leaving you to carry on with your voice call.

Because of its simplicity, **Point & Dial Voice Call** can be an excellent way to test your modem for the first time. To make this check, ensure that the modem power is on, and that you have the Framework II desktop on your screen. Press **Alt-F9**, which is the macro to display the Library Phone Book. Put the cursor on any field, press **Spacebar**, and type a phone number followed by **Return**. Since this is a test, choose a number that won't disturb anybody, such as the phone company's time service. Leave the cursor on the field you just typed in and select **Point & Dial Voice Call** from the Apps:Telecommunications submenu. If your modem has a speaker you will hear the computer dialing the phone. You may pick up the phone when Framework II prompts you to do so, or you may press the **Spacebar** as soon as you are convinced that the test has worked. The fact that the computer successfully dialed the phone tells you that the hardware is correctly set up. If the test failed and no number was dialed, then you will have to go back and compare your equipment to the suggested setup discussed earlier in this article. Check the settings on page 8-35 of *Using Framework II* for **Init String** and **Dial Prefix**. If your modem is not Hayes-compatible, chances are you will have to change these two menu settings. These options are also briefly discussed later in this article.

You should now feel confident that your equipment works as it should and that you are free to establish a connection to a computer in the outside world. To make such a connection, you first need to specify in Framework II the rules for talking to another computer. There are many ways to pass data from one system to another. Before you can call another system, you have to know the rules which are in effect on that system. These rules include the speed of data transfer, the format of the data, and other important parameters. Most of these options can be set from the Apps:Telecommunications:Setup:Setup Options submenu. This menu has three panels: Format, Modem, and Configuration. I will be discussing only the settings which are important for getting connected for the first time.

Format Panel

The first option of the Format panel is **Baud Rate**. This is the speed of communication, and is determined by capabilities of both your modem and the computer you will be communicating with. Generally, this setting will be 300, 1200, or 2400 baud. If you don't match speeds with the other computer, you will either fail to connect or you will see only "garbage" characters scrolling across your screen. The most commonly used speed is 1200 baud, and so is the default value for this setting.

The **Data Bits**, **Parity**, and **Stop Bits** options of the Format panel are all related to the format of information sent to and received from the remote system, and are often referred to collectively as "data word format." There are two frequently used data word formats:

Format	A	B
Data Bits	7	8
Parity	EVEN	NONE
Stop Bits	1	1

Format A is common for mainframe systems like Compuserve, The Source, and Telemail, whereas format B is typically used on private bulletin boards and other microcomputer-based systems. The decision of which data word format to use can often be made by reading the logon procedures for the particular service you are using.

Modem Panel

The Modem panel contains the settings for your particular modem. The default settings in this panel work well with Hayes- compatible modems, and some will only need to be changed if you have different equipment. **Init Modem String** is one of those choices, and is blank for a Hayes modem, but other modems may require that a special command be given at the start of operation. For instance, the Novation Smart-Cat requires the characters "%I" before any other commands may be sent to the modem. In that case, you would type the **%I** into the menu option for **Init Modem String**. **Prefix for Dialing** defaults to "AT DT", which is the Hayes Dial command that must precede a phone number. This setting will be the same any time you use Framework II telecommunications, but may be different for different hardware setups. If you have a pulse (rotary) dial phone line, for example, the Hayes Dial command would be "AT DP" instead.

The **Serial Port #** option on the Modem panel is another option you will always set the same. This refers to the serial port your modem is connected to. The value of this setting is one if the modem is on COM1:, and the value is two if the modem is on COM2:.

Assuming you have an auto-dial modem, the **Number to Dial** setting will be the one you use most frequently. Simply type in the phone number, including hyphens if you like (the hyphens will be ignored). For example, to dial an out-of-state number from an office phone that requires 9 for an outside line, you would put a number like this in the menu: 9-1-555-555-1212. It is possible that your office phone system requires that you wait for a dial tone after dialing 9. You can deal with this by taking advantage of the features of your auto-dial modem. A Hayes modem will pause for two seconds if it finds a comma within the phone number. To wait for two seconds after dialing 9, you would use a **Number to Dial** setting like this: 9,1-555-555-1212.

Configuration Panel

The Configuration panel contains settings you will not need to change very often. **Send Ctrl Chars** is the only setting you might want to change for your first few sessions. This setting is an ON/OFF toggle. It controls whether control characters will be sent to the remote system, or whether they will be used as Framework power keys. **Send Ctrl Chars** defaults to OFF, which means that **Ctrl-C**, for instance, will pull down the Create menu. Change the setting to ON if you will need to use control characters on the remote system. When **Send Ctrl Chars** is ON you will have to access the Framework II menus with the **Ins** key.

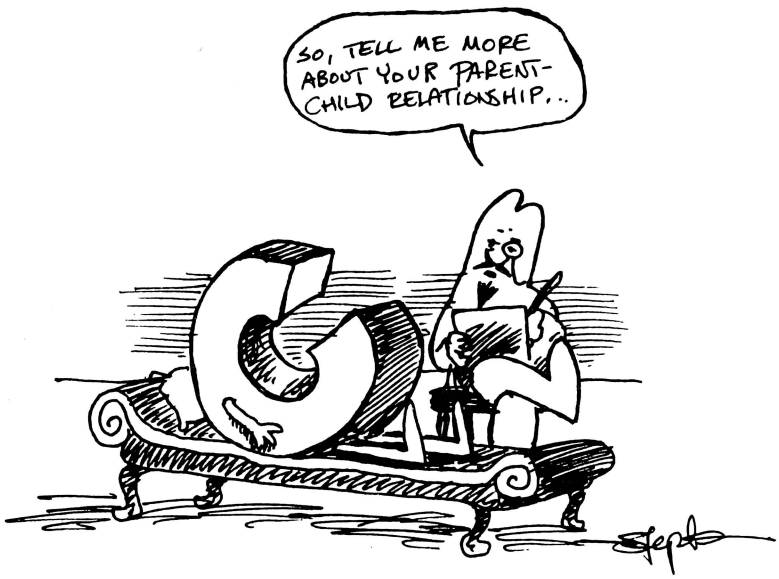
We have now scanned the Setup Options menu and, hopefully, helped you to change each of the important settings. Now press **Spacebar** and look at the Apps:Telecommunications:Setup menu. You can save all of the settings you just modified into a telecommunications setup file by selecting **Write Setup File** and typing a filename. All the information in the Telecommunications menus is saved to a file with a .TCM extension. This setup file will give you easy access to any service you use regularly. Simply **Load Setup File** and the settings you saved are installed in the menus again. Incidentally, if you want to start with a "clean slate" you can load the Telecomm setup file (Telecomm.TCM) from System Disk #2 and all settings will be restored to Framework II's defaults. For this reason, I recommend saving the options you just set into the Telecomm.TCM file so that **Point and Dial Voice Call** will work the next time you enter Framework II. Then, make one .TCM file for each different service you will be calling.

The complete hardware and software setup is now behind you, and you can select **Begin/Resume Session** from the Apps:Telecommunications:Setup menu to begin dialing. A Smartmodem with a speaker will allow you to hear the phone being dialed, the phone ringing, the carrier from the remote system, and, finally, a response carrier from your modem. At this point the speaker shuts off, and all further action will take place on your screen.

That should be enough tutoring for one issue. Next month, I will be continuing the "Telecommunications Primer" with a discussion of active Telecomm frames and how they can be used on the desktop and file transfers.

Arrays in dBASE TOOLS FOR C

by Robert Boies



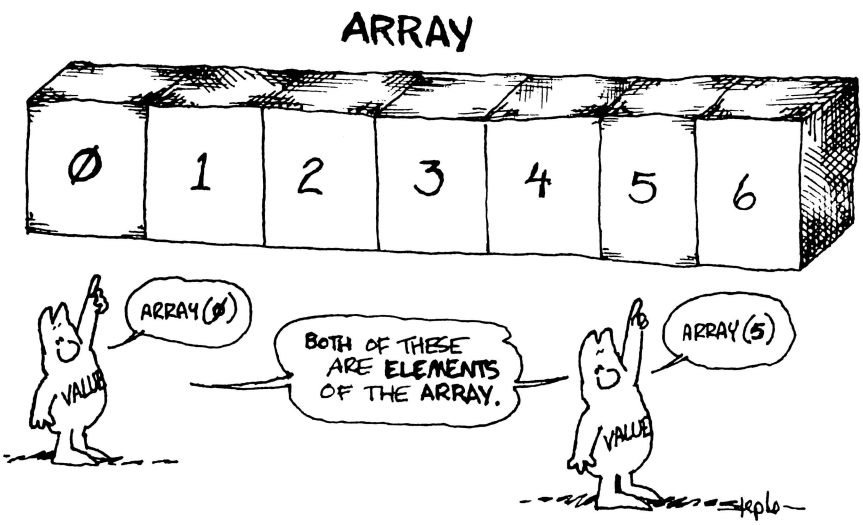
This article is the first in a series on interfacing the C language with dBASE III PLUS. Over the next several months I will be presenting a set of C routines that, when coupled with dBASE III PLUS, provide new and innovative ways to deal with some old problems, and new approaches to ideas that cannot be easily implemented in the dBASE III PLUS environment alone.

The mechanism I will use to link dBASE III PLUS to C is a new product called *dBASE TOOLS FOR C: The Programmer's Library*, written by J.T. Cooper & Company and published by Ashton-Tate. This package is a book-disk combination, a library of routines written in C. These routines are all available to the dBASE environment through assembly language interface. The dBASE III PLUS commands LOAD and CALL provide the operational link to the tools. The package supports Lattice, Microsoft, and Aztec C compilers. All external C source code in this series of articles will be presented in the Lattice format.

dBASE TOOLS comes ready to use with a full set of routines for financial functions, statistical functions, and array processing. Since most of the package's functions use the array management routines to perform their tasks, and arrays are new to the dBASE environment, this article will deal with ways to use arrays in dBASE III PLUS code.

What Is an Array?

Webster defines an array as "a regular and imposing grouping or arrangement . . . a group of elements forming a complete unit." In Mathematians and computer scientists further limit this classical definition by defining the group of elements to be a set which is typically addressed in memory, as opposed to file operations. Just as dBASE files are divided into records, arrays are divided into elements. Arrays are often referred to as "subscripted variables." A subscript is a simple integer number that points to the ordinal position of the element in the array. Just as individual dBASE records are referenced by the RECNO() function, array elements are referenced by subscripts. All arrays, no matter how complex, can be broken down into a finite collection of single elements. Each element has a fixed length and is maintained in a single dimension or vector. The name of the array used alone refers to the entire collection of elements, just as the name of the file refers to a collection of records. The following diagram illustrates how a subscript is used to point to a particular element of an array:



You pass to the array management routines the name of the array and a subscript number referencing a specific location in the array. The data stored in the requested location is then returned to a pre-defined dBASE III PLUS variable by the mechanisms provided within the dBASE TOOLS FOR C.

For the non-C oriented programmer, the article "Creating Bar Graphs," by Brian Evans in the August 1985 issue of ***TechNotes***, pages 5-11, provides further clarification of array concepts. In his article, Brian gives an example of passing dBASE the position in the pseudo array of memory variables, and dBASE returning the data stored in that element.

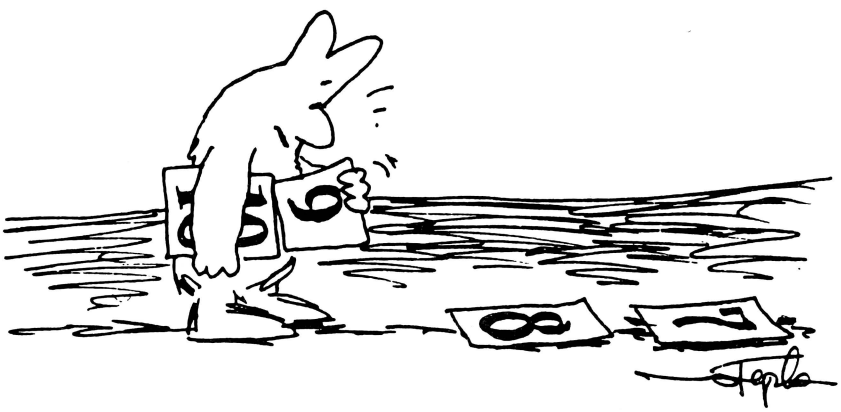
How Do We Use an Array?

Arrays are memory objects, as opposed to file objects, thus representing a new type of memory variable. The choice of what memory variable type to use now includes the array structure along with the standard character, date, logical, and numeric variable types.

To effectively use this new type of memory variable, follow the same basic rules as other memory variables with one important difference. You must declare and initialize the array in two steps. First create the array and then store information in its elements. Since the array is a collection of elements, addressing an element which has not been initialized may return unexpected results. The return value from an un-initialized element will be zero for numerics, null strings for character arrays, and whatever was in memory prior to array initialization for date arrays.

It is easy to see that array variables require special handling. To facilitate the use of the array data type, a set of primitives must be developed. A primitive is a piece of software which has the distinction of being data independent in its operation. The primitive is at the lowest level of hierarchy in any software system, and the operation of other routines depend on the primitive functions. The primitive itself is dependent only on the language in which it is coded, and performs one simple task. In the dBASE III PLUS, these primitives are often grouped together into a procedure file according to the action of the functions. The required primitives for array management must provide a common way to create an array, load an array with data, retrieve data from an array, and release the storage area set aside in memory for an array. In this article, we will define these primitive functions for interfacing with dBASE TOOLS FOR C.

When Do We Use an Array?



The array data type in other languages is the primary mechanism for data management in memory. In fact, programmers have been using a type of array in dBASE all along without realizing it. The storage of dBASE memory variables is done through an internal array mechanism. As mentioned previously, a valid analogy can be drawn between the construction of an array and the organization of the dBASE database file. The file can be viewed as an array of records, with the record numbers corresponding to the subscripts of the array. Files are an efficient way to manage data structures when the data is larger than memory can hold as one piece. In general, when the data can be stored in memory it makes more sense to load the data once, and access it in memory instead of from disk.

You can also use arrays to reduce the number of open files in the input/output system. The PC/MS-DOS operating system has a maximum of 20 files allowed open simultaneously. By the time the dot prompt is displayed, the maximum number of files allowed open is reduced to 15. A system of reasonable complexity will contain at least three to four data files and four to six index files. Thus, the total files available has been further reduced to between five and eight. Allowing for one format file, a procedure file, an alternate capture file, and a main program module further reduces the number of available files to between one and four. As the number of available files decreases, it becomes imperative to find memory-oriented solutions to file allocation problems. The array is such a solution.

In dBASE application programming, one way to benefit from arrays is use them as static lookup files. Such files could be parts inventories, tax tables, or sets of account numbers for general ledger systems. A database file used in this way has to be opened only once and the contents of the fields copied to an array. After the array is loaded, the database file can be closed and the data area freed for another database file.

Primitives

Creating an Array from a dBASE Database File

The first step in creating an array is to determine the total space needed by the array. For single dimensional arrays, the total number of elements in the array will be equal to the total number of records in the file. The second step is to determine what type of data the array is going to contain. With these two questions answered and a name for the array established, an array can be created.

The classical method of determining the amount of space needed for an array is to open the external data file containing the information, and count the total number of records. Once this count has been accomplished, the file is rewound to the beginning, and the array is created based on the total number of records counted. This is inefficient as the file will be read twice, once for counting the elements and again for actually loading the file. Fortunately for dBASE III PLUS programmers, the RECCOUNT() function has been included specifically for establishing the number of data records in our files.

The type of array created may be of any one of the dBASE variable types character, numeric, date, or logical. It is easy to see that simply selecting one of the types and coding the call to an array creation mechanism will solve this problem. But using this approach will make the code data-dependent, not exactly what primitives are made of. As stated previously, the primary goal of a set of primitives is to provide a series of data-independent mechanisms. Thus the design must first establish which field in the file will be loaded, and what data type this field is. The dBASE memory variable "field_nme," provides a symbolic mechanism for transferring any field name into the array creation routine as a parameter. When the variable is operated on with the TYPE() function, the type of array to create is returned. To establish even further data independence, the FIELD() function is employed so that the actual field name may be chosen at runtime. The code which follows is a free-standing procedure that will create an array based on a given field in a database file.

```
PROCEDURE Makearry
* Author ....: Robert Boies
* Date ......: April 1, 1986
* Note(s) ...: Procedure to create a new array.
```

```
*
PARAMETERS file_nme,area,field_nme,array_nme,rtrn_code
*
* file_nme     ::= Name of the dbf file
*                  {Type = C,length = 1 - 8}
*
* area         ::= Data area to use for file
*                  {Type = C,length = 1 or 2}
*
* field_nme    ::= Name of the field in the data file
*                  {Type = C,length = 1 - 10}
*
* array_nme    ::= Name of the array to create
*                  {Type = C,length = 1 - 8}
*
* rtrn_code    ::= Mechanism to return error status
*                  {Type = N,length = variable}
*                  VALUES = -2 == bad field name
*                  -1 == error on creation
*                  +<n> == number of recs, AOK
*
* ---Zero out the rtrn_code.
rtrn_code = 0
* ---Set up the data area and use the file.
SELECT &area
USE &file_nme
* ---Loop and locate the field we want to use.
fld_num = 1
DO WHILE (FIELD(fld_num) <> field_nme)
fld_num = fld_num + 1
ENDDO
* ---Check to see if an error has occurred.
IF "" = FIELD(fld_num)
rtrn_code = -2
ENDIF
* ---If we get this far without error all's OK
* ---so go ahead and set up the array as planned.
IF rtrn_code = 0
CALL cfunc WITH "CRARRAY " + array_nme + "," +;
                STR(RECCOUNT()) + ","+;
                TYPE(FIELD(fld_num))
ENDIF
* ---Set return flag to PUBLIC variable c_n_result and
* ---return it as a final error check.
rtrn_code = c_n_result
RETURN
* EOP Makearry
```

Loading an Array Element from a dBASE Data Field

The Makearry procedure does the job for which it was intended. It functions well for creating an array, but has no mechanism for loading information. For the time being the focus will be placed on single dimensional arrays created from a single field in a single data file. Most of the necessary information for loading the array was returned by the Makearry procedure. However, error checking of the file environment is not performed. For this reason the array should be loaded immediately after it is created, while still in the correct data area with the proper file in use. With these ideas in mind, the following dBASE code will load an array created by the Makearry procedure.

```
PROCEDURE Loadarry
* Author ....: Robert Boies
* Date ......: April 1, 1986
* Note(s) ...: Program to load array from a database file.
*
PARAMETERS field_nme,array_nme,num_parms,rtrn_code
*
* field_nme ::= Name of the field in the data file
*               {type = C,length = 1 - 10}
*
* array_nme ::= Name of the array to load
*               {Type = C,length = 1 - 8}
*
* num_parms ::= String of formal definition for
* numerics
*               {Type = C,length = variable
*               format = <Length>,<Decimals}
*
* rtrn_code ::= Mechanism to return error status
*               {Type = N,length = variable}
*               VALUES = -1 == error on load
*               +<n> == total elements, AOK
* ---Zero out the rtrn_code.
rtrn_code = 0
* ---Find out what kind of array we've got.
DO CASE
CASE TYPE(field_nme) ="D"
   * ---It's a date variable.
   key_expr = DTOC(&field_nme)
   c_d_result = CTOD("00/00/00")
CASE TYPE(field_nme) ="N"
   * ---It's a numeric variable.
   key_expr = STR(&field_nme,&num_parms)
CASE TYPE(field_nme) ="L"
   * ---It's a logical variable.
   key_expr = IIF(field_nme,"1","0")
OTHERWISE
   * ---It's a character variable.
   key_expr = &field_nme
ENDCASE
* ---Load the values into the array from the field one
* ---record at a time.
DO WHILE .NOT. EOF()
   CALL cfunc WITH "PUTARRAY " + array_nme + ", " +;
               STR(RECNO() - 1) + ", "+;
               key_expr
*
* ---Check the value of the PUBLIC variable c_status.
* ---If it's not 0, an error has occurred.
   IF c_status <> 0
      EXIT
   ENDIF
   * ---If no abend SKIP to the next record.
   SKIP
ENDDO
* ---STORE the status to the return code.
rtrn_code = c_status
RETURN
* EOP Loadarry
```

Retrieving the Contents of an Array Element

Now that there exists a mechanism for loading the array with data, construction of a retrieval routine can begin. One simple system model is the single keyed look-up table. Strictly speaking, a function can be considered a black box into which we pass a value and a new value pops out. The most simplistic model is the filter. This model is a program which accepts an input, filters it (perhaps testing the input against a known value), and returns a value. The function which follows assumes that the array has been previously built using Makearry and loaded with Loadarry. It also assumes that the data area selector pointer is positioned to the area number passed to the function.

```
PROCEDURE Seekarry
* Author ....: Robert Boies
* Date ......: April 1, 1986
* Note(s) ...: Procedure to find an element within a defined
*              array.
*
PARAMETERS key,array_nme,num_parms,elem_num,rtrn_code
*
* key       ::= Name of the field in the data file
*               {Type= variable,length= variable
*
* array_nme ::= Name of the array to create
*               {Type = C,length = 1 - 8}
*
* num_parms ::= String of formal definition for numerics
*               {Type = C,length = variable
*               format = <Length>,<Decimals}
*
* elem_num  ::= Number of the array element of data
*
* rtrn_code ::= Mechanism to return error status
*               and carries in the total elements
*               {Type = N,length = variable}
*               input VALUES = <n> number of elements
*               output VALUES = -1 error on retrieve
*               0 == AOK
*
* ---Zero out the Rtrn_Code and the Elem_num.
rtrn_code = 0
elem_num = 0
* ---Determine the variable type of the key.
vtype = TYPE(key)
DO CASE
CASE vtype = "D"
   * ---It's a date variable.
   key_expr = DTOC(&key)
   c_d_result = CTOD("00/00/00")
CASE vtype = "N"
   * ---It's a numeric variable.
   key_expr = STR(&field_nme,&num_parms)
CASE vtype = "L"
   * ---It's a logical variable.
   key_expr = IIF(key,"1","0")
OTHERWISE
   * ---It's a character variable.
   key_expr = &key
ENDCASE
* ---Search the values of the array one element at a time.
DO WHILE key <> c_&vtype._result
```

```
      CALL cfunc WITH "GETARRAY " + array_nme + ", " +;
                  key_expr + ", " + STR(elem_num)
      *
      * ---Check the value of the public variable c_status.
      * ---If it's not 0, an error occurred.
      IF c_status <> 0
         EXIT
      ELSE
         * ----Else increment elem_num.
         elem_num = elem_num + 1
      ENDIF
   ENDDO
   * ---Store the status to the return code.
   rtrn_code = c_status
   RETURN
   * EOP Seekarry
```

Removing an Array From the Runtime System

The final function in the list of necessary primitives for array manipulation frees memory when an array has outlived its usefulness. The following procedure, Freearry, performs this function.

```
PROCEDURE Freearry
* Author ....: Robert Boies
* Date ......: April 1, 1986
* Note(s) ...: Procedure to remove an array from memory.
*
PARAMETERS array_nme,rtrn_code
*
* array_nme     ::= Name of the array to create
*                   {type = C,length = 1 - 8}
*
* rtrn_code     ::= Mechanism to return error status
*                   {type = N,length = variable}
*                   output VALUES = -1 error on retrieve
*                   0 == AOK
*
* ---Initialize the status variable to 0.
c_status = 0
* ---Free the array from memory.
CALL cfunc WITH "FRARRAY " + array_nme
rtrn_code = c_status
RETURN
* EOP Freearry
```

Using ARRAYS

The Shoe Store

As an example of using arrays in an application, let us assume that we are programmers charged with the task of creating an inventory management system for a shoe store. The main inventory database file consists of the following fields:

FieldName	Type	Width	Dec
Size	Numeric	5	2
Style	Character	2	
Material	Character	2	
Width	Character	3	
Color	Numeric	2	0
On_hand	Numeric	4	0
Cost	Numeric	5	2
Retail	Numeric	5	2
Reorder	Numeric	2	0

The task is further narrowed in scope to produce a series of inventory reports giving information on available stock. Further, these reports must be printed in a format specified by the shoe industry. For example, if a record contains the following information:

Size 5.25
Style ZA
Material CO
Width EE
Color 50

the output for the report be:

Size 5 1/4
Style LOAFER
Material CORDUVAN
Width EE
Color OFFICERS PINK

To accomplish the translation without using dBASE TOOLS FOR C, four separate lookup files and index files would be required. Every time the manager wanted to produce the reports, all four files and their associated indexes would have to actively be in use. Thus, before actual processing of the data begins, eight of the fifteen available files would be used.

Most of these problems can be eliminated by building a set of arrays, initially storing the data from the four lookup database files into the array elements. When access to a particular element is requested, a simple sequential search of the array is performed. The array table is scanned for a match on the look-up value until one is located or the end of the array is reached.

Creating the ARRAY

The first step in the process is to create the arrays. To accomplish this task, use the primitives developed earlier in this article. All arrays in the dBASE TOOLS FOR C, and for that matter in C, are actually single dimensional, although they can assume the attributes of a multi-dimensional array. So, an array element can be referenced as an absolute or computed vector. For example, in dBASE TOOLS FOR C, the array element 15 can be referenced as the absolute vector, STR(15), or as the computed vector, STR(3 * 5). Note that the STR() function is used to reference elements in the array. The reason is that although dBASE TOOLS FOR C can work with numeric data, the DCT1.EXE module expects a single character string as its parameter. So, all non-character references must be converted to character type when passed.

The following dBASE III PLUS command lines create the arrays necessary for storing the contents of the lookup data file.

```
DO Makearry WITH "SIZE",10,"Size",rtrn_code
DO Makearry WITH "STYLE",9,"Style",rtrn_code
DO Makearry WITH "MATERIAL",8,"Material",rtrn_code
DO Makearry WITH "COLOR",7,Color,rtrn_code
```

Remember that arrays in C always start with an index of zero.

Putting Data into the Array

Assume that each of the look-up database files is composed of two fields, Key and Contents. We will put data into the arrays using the Loadarry procedure to load each element with the Key field, a blank space as a separator, and the Contents field. The following example demonstrates the general syntax.

```
SELECT <data area>
DO Loadarry with <field_nme>,<array_nme>,rtrn_code
```

Once the array is filled, the retrieval system can be placed in operation.

Getting Data from the Array

Retrieval of information from an array performs the equivalent of the SEEK or FIND command in dBASE III PLUS. All the code necessary to perform this type of operation is found in the primitive Seekarry. The search mechanism steps through the array, checking to see if the search argument is contained in the current element. If so, the search procedure is complete. If not, the element counter is incremented, and the searching operation is repeated. The following code demonstrates the technique of retrieving an element from an array. It assumes that the steps previously defined in this article have been followed, and that we are looking into the Color array for a given color, and the array has been previously loaded with values of:

Element	Contents
0	"AA RED"
1	"B3 TURQUOIS"
2	"AZ OFFICERS PINK"
3	"E2 GREEN TURTLE"

The key being searched for has the value "AZ", stored in variable "look_up". The code would be:

```
DO Seekarry with "AZ","Color",elem_num,rtrn_code
IF rtrn_code <> -1
shoe_color = SUBSTR(c_c_result,4,LEN(c_c_result) - 3)
ENDIF
```

The rest of the arrays in the system are handled similarly, with the exception of the Size array, which must be first searched for the numeric shoe size. If there is a match, then simply use GETARRAY to return the actual replacement string from the Str_size array. The only other difference for this construct is that the result of the numeric search will have to be retrieved from the default numeric variable, c_n_result before performing the translation using the Str_size array.

Conclusion

This article presents the initial techniques for creating and freeing arrays, loading array elements with data from a file, and finding a specified element in the array. These primitives demonstrate the concept of data independence. It is suggested that you use these primitives, or your own modified versions, as standard mechanisms for accessing and maintaining your arrays. Future articles demonstrating various array techniques will be based on how these primitives are used.

Corrigenda

(1) In the December 1985 issue of ***TechNotes***, on pages 56 and 57, Rob Lewis's article, "Using a RAM Disk with Extended Memory," omitted FWSETUP as one of the files that needs to reside on the RAM disk in order to start Framework II. If the FWSETUP file is not copied to the RAM disk, Framework II will ask you to insert System Disk #2. Additionally, if you do not copy LIBRARY.FW2 to the RAM disk, you will have an empty Library cabinet.

Operating dBASE III PLUS in a 256K Environment

by Gail Kulikowsky

dBASE III PLUS runs with PC-DOS version 2.xx if you have 256K installed memory in your computer. If you have a minimum of 384K of installed memory, dBASE III PLUS runs with PC-DOS version 3.xx as well as version 2.xx.

For 256K operation, two files are provided, CONFI256.SYS and CONFI256.DB, on System Disk #1. These two files set system parameters for maximum overall performance of dBASE III PLUS in a 256K environment.

The CONFI256.SYS file contains:

```
FILES = 20
BUFFERS = 4
```

Copy the CONFI256.SYS file to the root directory of the disk you boot DOS from or the directory set by the DOS COMSPEC command and rename it to CONFIG.SYS. If CONFIG.SYS already exists, modify the file to include FILES = 20 and BUFFERS = 4.

The CONFI256.DB file contains:

```
COMMAND = ASSIST
STATUS = ON
BUCKET = 1
GETS = 35
MVARSIZ = 3
HISTORY = 10
TYPEAHEAD = 10
```

Copy CONFI256.DB into be placed in the same directory as DBASE.COM and rename it to CONFIG.DB.

The CONFI256.DB settings reduce the memory-intensive parameters allowing dBASE III PLUS to work in a 256K environment. The smaller settings place the following limitations on dBASE III PLUS.

MVARSIZ is reduced to 3,000 bytes which is half the normal default size of 6,000 bytes. This reduces the amount of space available for memory variables and may cause incompatibilities with programs written with earlier versions of dBASE III. Refer to the SAVE and RESTORE commands for ways to optimize use of memory variable space.

BUCKET size is reduced to 1,024 bytes, half the normal default size. This reduces the amount of memory available for the PICTURE and RANGE options of the @...GET command, and may cause incompatibilities with programs written with earlier versions of dBASE III.

CREATE SCREEN is limited to one screen, with a maximum of 35 pending GETs. Format files created on machines with more memory may not work when moved to a 256K machine. The limitation of 35 pending GETs may also cause incompatibilities with programs written with earlier versions of dBASE III.

The RUN command is not supported on computers with 256K of memory. dBASE III PLUS requires 256K, the RUN command requires additional memory for COMMAND.COM plus the memory required by the external program. This is consistent with earlier versions of dBASE III.

The "Insufficient memory" message is an indication that some memory must be released to allow dBASE III PLUS to continue with the operation. When using dBASE III PLUS with 256K RAM this message may result from a combination of factors and varies with the size and number of database files, index files, format files, and the HISTORY and TYPEAHEAD buffer sizes.

If the "Insufficient memory" message is received:

1. Ensure that you are using the correct CONFIG.SYS and CONFIG.DB files, as listed above for 256K operation.
2. Close one or more open files and then proceed.

Increasing the amount of RAM in your computer will significantly enhance the performance of dBASE III PLUS.

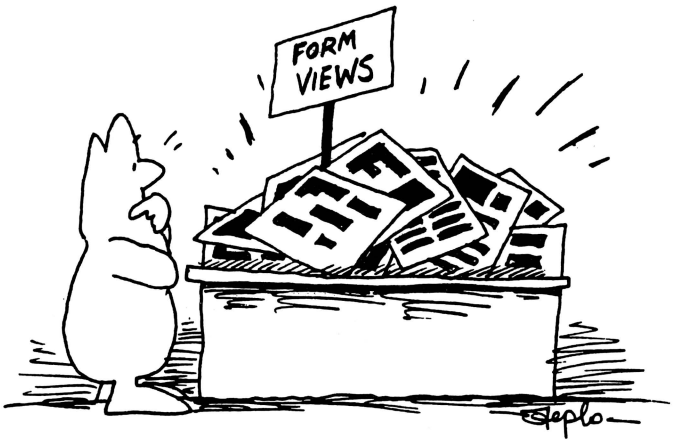
LISTing a Database File in a Vertical Form

by Stanley Ballanger

Many users have requested a way to LIST the contents of their database files in a vertical form rather than the horizontal table orientation of the LIST command. The following program, Listform.PRG, produces such a format from a database file in the currently SELECTed work area. It prints a vertical list of a database file, fields printing underneath one another. A typical example of the generated output looks like the following:

```
Database file :    C:Errors.DBF                          Page 1
================================================================

Record 1
Errorno...:    1
Message...: Non-existing file

Record 2
Errorno...:    2
Message...: Unassigned file no.

Record 3
Errorno...:    3
Message...: File open

Record 4
Errorno...:    4
Message...: End of file
```

The Listform program takes advantage of some of the new features in dBASE III PLUS. One of the new functions that is elemental to its operation is the FIELDS() function, which allows you to reference a field in a database file by its position in the file rather than by its name. You select a field by using a numeric variable as the parameter for the FIELDS() function. In addition, Listform uses the DBF() function to determine the database file in USE. The DBF() function returns the name of the database file in USE in the currently SELECTed work area or a null value, if there is no database file in USE.



To use the program, just type DO Listform and the current database file will LIST in the format of the example above. If there is no database file in USE, you will be prompted just as you are for the LIST command. Be aware that if you SET FIELDS or are using a view, you will still get all the fields of the current database file. The reason is that the FIELDS() function does not support the fields list specified by SET FIELDS or SET VIEW TO.

Listform.PRG

```
* Program ...: Listform.PRG
* Author ....: Stanley Ballanger
* Date ......: April 1, 1986
* Note(s) ...: Program to list a database file in a form format.
*
SET TALK OFF
* ---Check for an open database file in the current area.
IF "" = DBF()
   ACCEPT "No database is in USE. Enter file name: " TO dbf
   dbf = IIF("." $ dbf,LEFT(dbf,AT(".",dbf)-1),TRIM(dbf)+".DBF")
   IF "" = dbf .OR. (.NOT.FILE(dbf))
      ? "File does not exist."
      ?
      RETURN
   ENDIF
   USE &dbf
ENDIF
* ---Initialize local variables.
STORE 1 TO pgno, fcounter
lcounter = 99
pglength = 64
bmargin = 6
pgwidth = 75
indent = 12
linelen = pgwidth - indent
* ---Print the database file.
SET PRINT ON
DO WHILE .NOT. EOF()
   fcounter = 1
   * ---Check for new page.
   IF lcounter > pglength - bmargin
      * ---Print page heading.
```

```
      DO Newpage
   ENDIF
   * ---Process records.
   DO WHILE "" <> FIELD(fcounter)
      * ---Check for new page.
      IF lcounter > pglength - bmargin
         DO Newpage
      ELSE
         * ---Check for new record.
         IF fcounter = 1
            ?
            ? "Record ",LTRIM(STR(RECNO(),10))
            lcounter = lcounter + 2
         ENDIF
      ENDIF
      * ---Process field.
      fname = FIELD(fcounter)
      * ---Skip memo fields, can't deal with them.
      IF TYPE(fname) <> "M"
         * ---Print field title.
         ? SUBSTR(fname,1,1)+LOWER(SUBSTR(fname,2,10))
         ?? LEFT(".........",10-LEN(fname))+": "
         * ---Check for character strings that need to word wrapped.
         IF TYPE(fname) = "C"
            IF LEN(TRIM(&fname)) > linelen
               DO Wrap WITH &fname
            ELSE
               ?? TRIM(&fname)
               lcounter = lcounter + 1
            ENDIF
         ELSE
            ?? &fname
            lcounter = lcounter + 1
         ENDIF
      ENDIF
      fcounter = fcounter + 1
   ENDDO
   SKIP
ENDDO
SET PRINT OFF
SET TALK ON
RETURN
* EOP Listform.PRG
```

```
* Program ...: Wrap.PRG
* Author ....: Stanley Ballanger
* Date ......: April 1, 1986
* Note(s) ...: Program to wrap long strings for Listform.PRG
*
PARAMETERS fld
firstline = .T.
start = 1
strlen = LEN(fld)
line = SUBSTR(fld,start,linelen)
* ---Wrap the long field until there is less than a line
* ---to print.
DO WHILE LEN(line) = linelen
   * ---Walk backwards to find the last blank character
   * ---in the line.
```

```
   cnt = 1
   DO WHILE RIGHT(line,cnt) <> " "
      cnt = cnt + 1
   ENDDO
   * ---Assign the next line position pointer.
   start = start + linelen - IIF(cnt > 1,cnt,0)
   * ---Check for new page.
   IF lcounter > pgwidth - bmargin
      DO Newpage
   ENDIF
   * ---Print the line, but check first to see if
   * ---it the first line of the record.
   IF firstline
      ?? LTRIM(LEFT(line,linelen - cnt + 1))
      firstline = .F.
   ELSE
      ? SPACE(indent)+LTRIM(LEFT(line,linelen - cnt + 1))
   ENDIF
   lcounter = lcounter + 1
   * ---Get the next line.
   line = SUBSTR(fld,start,;
               IIF(start < strlen - linelen,linelen,strlen-start))
ENDDO
* ---Check for new page.
IF lcounter > pgwidth - bmargin
   DO Newpage
ENDIF
* ---Print last line.
? SPACE(indent) + LTRIM(RIGHT(fld,strlen - start + 1))
lcounter = lcounter + 1
RETURN
* EOP Wrap.PRG
```

```
* Program ...: Newpage.PRG
* Author ....: Stanley Ballanger
* Date ......: April 1, 1986
* Note(s) ...: Program to print page top for Listform.PRG
*
EJECT
blanks = pgwidth - (26 + LEN(DBF() + LTRIM(STR(pgno,10))))
? "Database file : ",DBF(),SPACE( blanks ),"Page ",LTRIM(STR(pgno,10))
? REPLICATE("=",pgwidth - 1)
?
? "Record ",LTRIM(STR(RECNO(),10)),"(Cont'd)"
lcounter = 4
pgno = pgno + 1
* EOP Newpage.PRG
```

An Example of Catalog Management

by Quinn Wildman

One of the new features of dBASE III PLUS is the catalog. The catalog is database file that contains information about database files and associated files such as views, formats, indexes, queries, memo, LABEL and REPORT FORMs.

A catalog is opened in work area 10 with the SET CATALOG TO file/? command. Files accessed after a catalog file has been opened are recorded into the catalog until the catalog is closed or SET OFF. Each file is recorded only the first time it is used. The structure of the catalog database file is as follows:

Field	Field Name	Type	Width	Dec
1	Path	Character	70	
2	File_name	Character	12	
3	Alias	Character	8	
4	Type	Character	3	
5	Title	Character	80	
6	Code	Numeric	3	0
7	Tag	Character	4	

For a description of each of these fields see Using section of the *dBASE III PLUS Reference Manual* pages 3-15 and 3-16.

There also is a catalog of catalogs. Its name is Catalog.CAT and has the same file structure as the catalog. Entries are entered in the same manner as the regular catalog. Whenever you SET CATALOG TO file, it first looks to see if that file exists in Catalog.CAT and if it doesn't a new record is created for the new catalog entry.

The following four programs, Main.PRG, Database.PRG, Options.PRG and Display.PRG. comprise a small file management system that uses the catalog as its basis. It uses the query (?) clause feature of catalogs in combination with ON ERROR to allow you to control access to and manipulate the contents of a given catalog. The query (?) option works by allowing you to select from a list of files presented in a pull-down type of menu selector. A file descriptor title appears in a message box located in the lower right of the screen as you scroll up and down the file list.

This system allows you first, to start at the level of the system catalog, Catalog.CAT and select the catalog you wish to work from. Second, you select the database file you want within. Third, you select an index file and or a query file to order and filter the set of records you want to work with. And lastly, you output or manipulate your data. The system operates on the hierarchy of data structures in the following way.

Main.PRG ---> Database.PRG ---> Options.PRG ---> Display.PRG

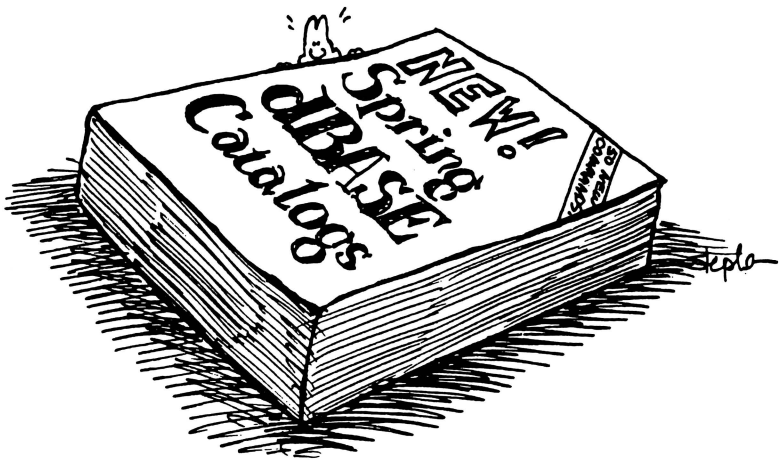
Catalogs Databases Indexes Reports Queries Labels

At each level of the system, you can add files to the catalog. You can as well ERASE files and this option is located on the Database.PRG menu.

To run the program, type DO Main from the dot prompt.

MAIN

```
* Program ...: Main.PRG
* Author ....: Quinn Wildman
* Date ......: April 1, 1986
* Note(s) ...: Main driver program for the catalog management system.
*
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SCOREBOARD OFF
level = 1
ON ERROR DO Err WITH level
DO WHILE .T.
   CLEAR
   selection = 1
   SET COLOR TO N/W
   @  0,0 TO 8,20 DOUBLE
   SET COLOR TO W/N
   @  2,1 TO 2,19
   @  1,2 SAY "CATALOG"
   @  3,2 SAY "1. Select "
   @  4,2 SAY "2. Add entry"
   @  5,2 SAY "3. Quit "
   @  7,2 SAY "Enter Selection" ;
          GET selection PICTURE "#" RANGE 1,3
   READ
   DO CASE
   CASE selection = 1
      SET CATALOG TO ?
      SELECT 10
      IF "" <> DBF()
         * ---This means escape was not hit in response to
         * ---choosing catalog.
         SELECT 1
         DO Database
      ENDIF
   CASE selection = 2
      CLEAR
      name = SPACE(8)
      @  2,0 SAY "Enter new catalog name";
             GET name
      READ
      SET CATALOG TO &name
   CASE selection = 3
      EXIT
   ENDCASE
ENDDO
ON ERROR
CLEAR ALL
SET BELL ON
SET TALK ON
* EOP Main.PRG
```



DATABASE

```
* Program ...: Database.PRG
* Author ....: Quinn Wildman
* Date ......: April 1, 1986
* Note(s) ...: Program to maintain database files.
*
DO WHILE .T.
   selection = 1
   CLEAR
   SET COLOR TO N/W
   @ 3,16 TO 12,41 DOUBLE
   SET COLOR TO W/N
   @  5, 17 TO 5,40
   @  4, 18 SAY "DATABASE"
   @  6, 18 SAY "1. Choose Database "
   @  7, 18 SAY "2. Add Database "
   @  8, 18 SAY "3. Delete Files "
   @  9, 18 SAY "4. Return TO Main Menu"
   @ 11, 18 SAY "Enter Selection";
          GET selection PICTURE "#" RANGE 1,4
   READ
   name = SPACE(8)
   DO CASE
   CASE selection = 1
      level = 9
      USE ?
      level = 1
      IF "" <> DBF()
         * ---This means Esc was not pressed.
         DO Options
      ENDIF
   CASE selection = 2
      CLEAR
      DIR
      @ ROW()+1,00 SAY "Enter name of database file " GET name
      READ
      level = 2
      USE &name
      level = 1
   CASE selection = 3
```

```
      level = 16
      ERASE ?
      level = 1
   CASE selection = 4
      EXIT
   ENDCASE
ENDDO
* EOP Database.PRG
```

OPTIONS

```
* Program ...: Option.PRG
* Author ....: Quinn Wildman
* Date ......: April 1, 1986
* Note(s) ...: Program to allow access to indexes and queries.
*
DO WHILE .T.
   CLEAR
   selection = 1
   SET COLOR TO N/W
   @  0,22 TO 14,48 DOUBLE
   SET COLOR TO W/N
   @  2,23 TO 2,47
   @  1, 24  SAY "OPTIONS"
   @  3, 24  SAY "1. Select Index "
   @  4, 24  SAY "2. Add Index to Catalog "
   @  5, 23  TO 5,47
   @  6, 24  SAY "3. Select Query "
   @  7, 24  SAY "4. Add Query to Catalog "
   @  8, 23  TO 8,47
   @  9, 24  SAY "5. Goto View Data Menu"
   @ 10, 24  SAY "6. Goto Previous Menu"
   @ 11, 24  SAY "7. Goto Main Menu"
   @ 13, 24  SAY "Enter Selection" ;
             GET selection PICTURE "#" RANGE 1,9
   READ
   name = SPACE(8)
   DO CASE
   CASE selection = 1
      level = 11
      SET INDEX TO ?
      level = 1
      * ---First consider if Esc has been hit.
      indx = NDX(1)
      IF "" <> NDX(1)
         * ---Procedure to open all other database files.
         datab = DBF()
         SELECT 10
         LOCATE for file_name = datab
         codev = code
         LOCATE for code = codev
         macro = indx
         cnt = 1
         DO WHILE FOUND() .AND. cnt < 8
            IF file_name <> indx
               macro = index + ","+ file_name
               cnt = cnt + 1
```

```
            ENDIF
            CONTINUE
         ENDDO
         SET INDEX TO &macro
      ENDIF
   CASE selection = 2
      CLEAR
      @  3,0 SAY "Enter name of index file ";
            GET name
      READ
      level = 4
      SET INDEX TO &name
      level = 1
   CASE selection = 3
      level = 12
      SET FILTER TO FILE ?
      level = 1
   CASE selection = 4
      CLEAR
      @  3,0 SAY "Enter name of query file " GET name
      READ
      level = 5
      SET FILTER TO FILE &name
      level = 1
   CASE selection = 5
      DO Display
   CASE selection = 6
      RETURN
   CASE selection = 7
      RETURN TO MASTER
   ENDCASE
ENDDO
RETURN
* EOP Option.PRG
```

DISPLAY

```
* Program ...: Display.PRG
* Author ....: Quinn Wildman
* Date ......: April 1, 1986
* Note(s) ...: Program to perform data manipulation operations.
*
DO WHILE .T.
   CLEAR
   selection = 1
   SET COLOR TO N/W
   @  0,22 TO 17,50 DOUBLE
   SET COLOR TO W/N
   @  2, 23 TO 2,49
   @  1, 24  SAY "VIEW DATA"
   @  3, 24  SAY "1. Reports "
   @  4, 24  SAY "2. Add Report to Catalog "
   @  5, 23  TO 5,49
   @  6, 24  SAY "3. Labels "
   @  7, 24  SAY "4. Add Label to Catalog"
   @  8, 23  TO 8,49
   @  9, 24  SAY "5. Edit Data"
```

```
@ 10, 24  SAY "6. Append Data"
@ 11, 24  SAY "7. Browse Data"
@ 12, 23  TO 12,49
@ 13, 24  SAY "8. Goto Previous Menu"
@ 14, 24  SAY "9. Goto Main Menu"
@ 16, 24  SAY "Enter Selection" ;
        GET selection PICTURE "#" RANGE 1,9
READ
pr = .T.
name = SPACE(8)
DO CASE
CASE selection = 1
   CLEAR
   @  3,0 SAY "Do you want to print?" ;
          GET pr PICTURE "Y"
   READ
   IF pr
      SET PRINT ON
   ENDIF
   level = 14
   REPORT FORM ?
   level = 1
   SET PRINT OFF
CASE selection = 2
   CLEAR
   @  3,0 SAY "Enter report name" ;
          GET name
   @  5,0 SAY "Do you want to print? " ;
          GET pr PICTURE "Y"
   READ
   IF pr
      SET PRINT ON
   ENDIF
   level = 7
   REPORT FORM &name
   level = 1
   SET PRINT OFF
CASE selection = 3
   CLEAR
   @  3,0 SAY "Do you want to print? " ;
          GET pr PICTURE "Y"
   READ
   IF pr
      SET PRINT ON
   ENDIF
   level = 15
   LABEL FORM ?
   level = 1
   SET PRINT OFF
CASE selection = 4
   CLEAR
   @ 3,0 SAY "Enter label name" ;
         GET name
   @  5,0 SAY "Do you want to print? " ;
          GET pr PICTURE "Y"
   READ
   IF pr
      SET PRINT ON
   ENDIF
   level = 8
   LABEL FORM &name
   level = 1
   SET PRINT OFF
```

```
   CASE selection = 5
      CLEAR
      rec = RECCOUNT()
      @  3,0 SAY "Enter record number to edit " ;
             GET rec RANGE 1,RECCOUNT()
      READ
      EDIT rec
   CASE selection = 6
      APPEND
   CASE selection = 7
      BROWSE
   CASE selection = 8
      RETURN
   CASE selection = 9
      RETURN TO MASTER
   ENDCASE
ENDDO
RETURN
* EOP Display.PRG
```

ERR

```
* Program ...: Err.PRG
* Author ....: Quinn Wildman
* Date ......: April 1, 1986
* Note(s) ...: Program to deal with error conditions.
*
PARAMETERS level
IF level = 1
   * ---Should never be executed.
   @  3,0 SAY MESSAGE()+STR(ERROR())
   DISPLAY MEMORY
   ON ERROR
   CLEAR ALL
   SET BELL ON
   SET TALK ON
CLEAR ALL
   CANCEL
ELSE
   IF ERROR()=1
      * ---File was not found.
      cr = .T.
      @  3,0 SAY MESSAGE()+" Do you wish to create it?" ;
             GET cr
      READ
      IF cr
         DO CASE
         CASE level = 2
            CREATE &name
         CASE level = 3
            CREATE VIEW &name
         CASE level = 4
            expr = SPACE( 100 )
            @  5,0 SAY "Enter index expression" ;
                   GET expr
            * ---Take in consideration that expression may be illegal
            READ
```

```
            INDEX ON &expr TO &name
         CASE level = 5
            CREATE QUERY &name
            level = 99
         CASE level = 7
            CREATE REPORT &name
         CASE level = 8
            CREATE LABEL &name
         CASE level = 99
            RETURN
         ENDCASE
      ELSE
         RETURN
      ENDIF
   ELSE
      IF ERROR() = 53
         CLEAR
         DO CASE
         CASE level = 9
            @  3,0 SAY "No database files in Catalog, add some first"
         CASE level = 10
            @  3,0 SAY "No view files in Catalog, add some first."
         CASE level = 11
            @  3,0 SAY "No index files in Catalog, add some first."
         CASE level = 12
            @  3,0 SAY "No query files in Catalog, add some first."
         CASE level = 13
            @  3,0 SAY "No format files in Catalog, add some first."
         CASE level = 14
            @  3,0 SAY "No report files in Catalog, add some first."
         CASE level = 15
            @  3,0 SAY "No label files in Catalog, add some first."
         ENDCASE
         WAIT
      ELSE
         IF ERROR() = 89 .AND. level = 16
            CLEAR
            @  3,0 SAY "Cannot delete open file , "+;
                       "use another file or return to main menu"
            WAIT
         ENDIF
   ENDIF
      ENDIF
   ENDIF
ENDIF
* EOP Err.PRG
```

A Guided Tour of the Library Cabinet

by Paul Bukauskas

Perhaps the first detail that will catch the eye of a first-time Framework II user is the Library cabinet residing unobtrusively on the desktop. As its label implies, the Library is a collection. In the same way that your neighborhood public library is a collection of books, the Framework II Library cabinet is a collection of custom tools which you develop to assist you in using Framework II. This article will take you on a tour of the Library you receive with Framework II, and highlight some of the powerful tools that are provided for you. As you gain confidence, you can customize these tools to your own needs and make Framework II into your personal document handler.

The Library is much more than a convenient container for macros and abbreviations; it is a vehicle that liberates you from needless repetition. Each frame, or "tool," in the Library is a part of a pool of resources that can be accessed from anywhere in Framework II. By placing a new function, macro, template, or abbreviation into the Library, you are essentially extending Framework II itself and producing your own personal version of the product.

As you might expect, everything in the Library is a frame and so the most natural way to organize is to use an outline. That is the way the Library appears when you first use Framework II, and a part of it looks like this:

```
> Printing Templates
   - Pica   ; Memo in pica type.
   - Elite  ; Business letter in elite type.
> Abbreviations
> Names
   - AT     ; Ashton-Tate
   - PCB    ; Paul C. Bukauskas
> States
   - CA     ; California
   - NY     ; New York
```

All of the printing templates are grouped under one descriptive heading, and all the abbreviations under another. These nested outline levels are used in the Library simply to group your tools and to make maintaining them easier. If you had no grouping at all, each of the tools would still work properly. You can also think of the Library as being similar to the PC/MS-DOS SET PATH command, because if Framework II does not find a referenced frame on the desktop, it will search the Library.

Your Framework II Library is organized into five major areas: Macros, Abbreviations, User-defined functions, Printing templates, and Assumptions. However, you can organize the Library any way you want, by deleting some tools and expanding others.

Macros

Let's begin our tour of the Library with macros. Macros play a major role in Framework II, and you will probably work with them more than any of the other Library tools. Ten handy macros are provided to get you started. The first two, {ALT-F1} and {ALT-F2}, are used for Library maintenance. Pressing the **ALT-F1** key displays the Library contents, saving you the keystrokes necessary to navigate to the Library cabinet and zoom the cabinet open. The {Alt-F2} macro saves the Library to disk. In addition, Framework II will always prompt you to save the Library at the end of a session if any changes have been made.

The next pair of macros, {ALT-F3} and {ALT-F4}, type the current date or time in the current document. These are handy when you want to quickly insert either the date or time into a word frame, frame label, or cell. Practical application of these macros include time-stamping documents or programs before saving, and placing the current date in a letter.

When you become more familiar with Framework II, and work consistently with large documents, {Alt-F5} will be a valuable tool. This macro displays the amount of desktop memory still available to you. It is easy to manage your desktop documents by keeping close track of the remaining memory, and removing documents from the desktop when things get crowded.

{ALT-F6} deserves close examination. When pressed, this macro displays a menu at the bottom of the screen, offering you the options of a memorandum, company letter, and personal letter. Once a menu item is selected, the macro creates a "skeleton" document of the chosen format. In the company letter, for example, the senders name, title, company name, company address, and current date are all automatically typed into the letter. All that remains is for you to type the receiver's address and the body of the letter.

Similarly, the memorandum and personal letter options create basic forms. The information for the templates is found in the Library. You can add to the Library frames containing your name, your company address, and other commonly used phrases. Although normally found in the Assumptions area, these frames could appear anywhere in the Library.

{Alt-F7} and {Alt-F8} are cut and paste macros. In many instances these two macros are preferred over the **MOVE (F7)** and **COPY (F8)** commands. Whereas **COPY (F8)** provides instant copies of characters, cells, or frames, the cut and paste macros can store and retrieve words or formulas. This differs from the **COPY (F8)** and **EXTEND SELECT (F6)** key combination because the **COPY (F8)** key is unable to copy formulas from one formula area to another. Nor will **COPY (F8)** copy text from a word frame to a formula, whereas the cut and paste macros can move text to or from formulas easily. Another difference is that once text is cut to the Cut Paste Buffer with **Alt-F7**, that same text may be repeatedly pasted into documents with **Alt-F8**, giving you an easy to use "boilerplate" technique. This "clipboard" feature is unavailable with a normal copy command.

The last two macros are also a working pair. The {ALT-F9} macro quickly displays the Library's Phone Book database. When {Alt-F9} is pressed the database appears full screen, allowing you to add or delete people from the list. At this point, you can also place the cursor on a phone number and press **Alt-F10** to dial the number for a voice call. The {ALT-F10} macro will dial a phone call with a phone number that is selected in a word frame, a spreadsheet, or a database. The phonebook feature can be used at any time—allowing you to "pop-up" your phone list, dial a call, and return to your work by pressing **Esc**.

Abbreviations

The abbreviation feature of Framework II is a great way to save keystrokes on common words or phrases. Abbreviations, like macros, are recorded via the Create pull-down menu. The steps to create an abbreviation are as follows:

1. Make sure you are in a word frame.
2. Press **Ctrl-C-M** to select the Create menu and the **Macro/Abbreviation** option.
3. Type the abbreviation. For example "AT". Once the abbreviation is typed, press **Return**.
4. Next, type the full text that the abbreviation is to represent. In this case, type "Ashton-Tate". To finish recording the abbreviation, press **Ctrl-Break**.
5. To use the abbreviation, type the abbreviated spelling and press **Alt-Backspace**. Note that pressing **Ctrl-Backspace** will erase the abbreviation you just typed.

Because abbreviations are not limited to a single word or phrase, a few characters can represent a commonly used paragraph, thus saving numerous keystrokes. One not so obvious application involving abbreviations is in FRED programming. Some common FRED functions can easily be abbreviated. An example is **@performkeys**. The abbreviation "pf" could stand for the full function name, and you could type "pf" **Alt-Backspace** to enter **@performkeys** into the formula, thus saving several keystrokes.

User-Defined Functions

This section of the Library is where you can keep new functions which you have written to customize the Framework II FRED language. Three pre-written user-defined functions come with Framework II. These functions, when used in a cell, return the weekday, hour, or minute. To add new functions, you simply create an empty frame and put your function in the formula area. For example, the following function, called Hex, converts a decimal number (base ten) to its hexadecimal (base sixteen) equivalent. To add Hex to the Library, create an empty word frame, label it Hex, and enter the following formula behind the frame.

```
; Function HEX

; by Olivier Biggerstaff

; Accepts a decimal number and returns its hex equivalent.

@local(out,number,byte1,byte2,string,char1,char2,char3,char4),
@if( @iserr ( @item1 ) = #FALSE,
   @list(
       number:=@item1,
       @if( @and ( number >= 0, number <= 65535 ),
            @list(
             string := "0123456789ABCDEF",
             byte1  := @mod( number,256 ),
             byte2  := @int( number/256 ),
             char1  := @mid( string, @int( byte2/16 ) +1, 1 ),
             char2  := @mid( string, @mod( byte2,16 ) +1, 1 ),
             char3  := @mid( string, @int( byte1/16 ) +1, 1 ),
             char4  := @mid( string, @mod( byte1,16 ) +1, 1 ),
             out    := char1 & char2 & char3 & char4
            ),
       )
 ),
),
@return(out)
```

Once in the Library, the Hex function can be used in the formula of any frame or cell. The parameter is passed to the function within parentheses. For instance, @Hex(100) would return 64. Similarly, Dsum and Dcount, found on the Framework II Sample Program disk, can be placed into the Library and used like any other Framework II functions. User-defined functions are a way to make more powerful and personal spreadsheets.

Printing Templates

A printing template is simply a frame that has special print attributes which you use frequently. You set these attributes using the normal Print pull-down menu. Then, for future documents, rather than setting the print options again, you just indicate the Library template name in the Print menu.

Printing templates are useful because they make use of print attribute inheritance. Inheritance means that sub-frames inherit the attributes of their containing frame. In the outline below, each frame inherits the attributes of Main Heading. If Main Heading is changed, frames 1.1, 1.2, and 1.3 similarly change. Subheading 1.2 is able to override Main heading, but it passes these new changes on to 1.2.1. When a subheading has different attributes than its containing frame, then the **Reveal Type** command in the Frames pull-down menu will show a /p after the frame name.

> 1 Main Heading
  - 1.1 Main head
> 1.2 individual setting /p
  - 1.2.1 individual settings (W)
  - 1.3 Main heading (W)

A printing template allows the user to quickly override inherited attributes. This is especially useful in the case where frames are copied into an outline frame. A frame copied to another frame keeps its own print attributes, and so can continue to use the attributes of a Library template in its new location. For more information on printing templates, see page 7-42 of *Using Framework II.*

A number of printing templates are already available in the Library. In particular, there are the Pica and Elite templates intended for either the IBM Proprinter or the IBM Graphics Printer. These templates provide headers and footers as well as the proper control codes to invoke their functions.

Assumptions

As mentioned earlier in this article, Framework II automatically looks for referenced frames in the Library. You could put a frame containing the current interest rate or rate of exchange into the Assumptions section. Then each time Interest rate was used in a formula, the current figure in the Library would be used. For instance, the formula:

B22 * Interest

would multiply the value of cell B22 with the value of the frame labeled Interest in the Library.

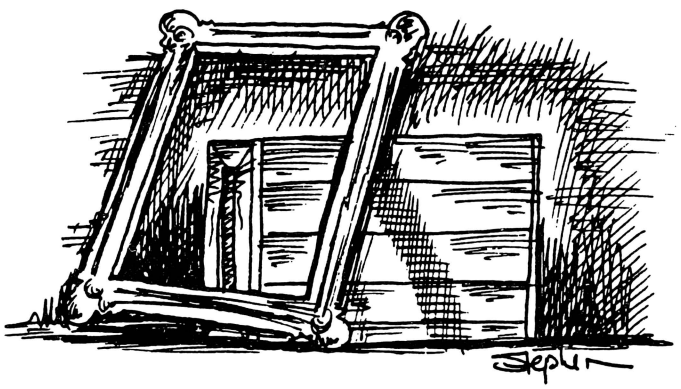
The {Alt-F6} macro also makes the assumption that the user's name, address, and company name are found in the Library Assumptions. Change the information in these frames so it is appropriate for the kind of letters you will be writing.

The Library cabinet provides you with many possibilities, and just a few of them have been discussed here. Remember that you can organize the Library in outline form for clarity, although the precise order and depth of frames is of no consequence. And best of all, the Library can grow in size and complexity commensurate with your skill level. As you become more familiar with it, the Library will become one of the most valuable features available in Framework II.

How to Use Boxes in Word Frames

by Gail Kulikowsky

Have you ever found yourself wishing you could set off some of the information in your document by placing it in a box? Or perhaps you wanted to print the frame border itself without having to do a **Shift-PrtSc** and edit out the rest of the desktop.



Boxes can be drawn manually in a word frame with the **Alt** key and the numeric keypad. However, if you draw them often or if the boxes are large, this becomes very tedious. In the interest of automation, we will create a macro to accomplish this task, and give ourselves more time for other productive pursuits.

To set up this macro, go into the Library and create an empty word frame. Label this frame {ALT-B}. Go into the {ALT-B} frame and create two more empty word frames. Label these Box and Nextline, respectively. Enter the following formulas into the indicated frames.

{Alt-B}:

```
; {ALT-B} Library macro.
; by Gail Kulikowsky.

@Box
```

Box:

```
; Draw a Box Macro.

@local(linelen, iv, hlen, vlen, vline, col, lastcol),
```

```
; Determine the Screen Position and Set Typeover ON
col := @panel1-1,
@if(@sense(#ET) = #false,
     @list(@echo(#OFF),
          @performkeys("{Ctrl-E}T"),
          @echo(#ON)
     )
),

; Get values for horiz. and vert. size,
; and set up a function for line length.

hlen := @integer(@value(
     @inputline("Enter the horizontal width (1-79)"))-2),
vlen := @integer(@value(
     @inputline("Enter the vertical length (1-54)"))-1),
lastcol := @panel1 + @value(hlen),

@if(@len(hlen)=2,
     linelen := "@performkeys(""{Ctrl-" & @mid(hlen,1,1) & "}{Ctrl-" &
        @mid(hlen,2,1) & "}"")",
     linelen := "@performkeys(""{Ctrl-" & hlen & "}"")";)
),


; Draw top line

@performkeys(@chr(218)),
@linelen,
@performkeys(@chr(196)),
@performkeys(@chr(191)),
@nextline(col, lastcol),

; Draw body lines

iv := 1,
vline := "@performkeys(@chr(179))",
@while(iv < @value(vlen),
     @vline,
     @linelen,
     @performkeys(" "),
     @vline,
    @nextline(col, lastcol),
     iv := iv + 1,
),

; Draw bottom line

@performkeys(@chr(192)),
@linelen,
@performkeys(@chr(196)),
@performkeys(@chr(217)),
@nextline(col, lastcol)
```

Nextline:

```
; Nextline.
; by Gail Kulikowsky.

@local(w, x, y, z, back),
w := @panel1,
x := @panel2,

 ; Store value of starting column

z := @integer(@item1),
@if(@len(z)=2,
     back := "@performkeys(""{Ctrl-" & @mid(z,1,1) & "}{Ctrl-" &
          @mid(z,2,1) & "}"")",
     back := "@performkeys(""{Ctrl-" & z & "}"")";)
),
@performkeys("{dnarrow}"),
y := @panel2,
@if(x=y,
     @list(
          @performkeys("{Return}"),
          @back,
          @performkeys(" ")
     ),
     @if(@item2 > w,
          @list(
               @performkeys("{Home}"),
               @back,
               @performkeys("{Rightarrow}")
          ),
          @list(
               @performkeys("{Home}"),
               @back,
               @performkeys(" ")
          ) ; end list.
     )      ; end if.
)           ; end if.
```

To use the **Alt-B** macro, go into a word frame, put the cursor where you want to draw the box, and press **Alt-B**. You will be prompted for the width and the height and then the macro will draw the box. Be sure to draw the box first and then enter the text, as the box will overwrite any text in its path.

The IBM characters I have used for the box border are:

Character	Displays	Character	Displays
@chr(218)	┌	@chr(179)	│
@chr(196)	─	@chr(192)	└
@chr(191)	┐	@chr(217)	┘

These characters will draw a single-line box. If you prefer some other type of box border you can substitute the appropriate ASCII codes into the formula of Box. Consult your computer manual for available characters.

To print the box as it looks on the screen, you will need an IBM graphics printer or a printer with the IBM character code set.

Using the graphics characters available with the **Alt** key and the numeric keypad opens up a whole new world of creative things you can do with a document. Hopefully this small sample will be only the beginning for you.

USAGE TIPS

by Joseph Perret

dBASE III PLUS

@...SAY

When sending control codes to the printer with @...SAY, control codes are counted as characters by dBASE III PLUS although these characters are not printed by the printer. This causes the print position coordinates for following @...SAY commands to the same line to be offset left by the number of control code characters sent to the printer. For instance, if one control character is sent to row 5, the column coordinate on row 5 will be less by one.

```
@ 5,0 SAY CHR(15)
@ 5,12 SAY "Test"
```

will print "Test" starting at column 11. The control codes for the following example are for the C-Itoh 8510A printer.

```
SET DEVICE TO PRINT
@ 5,5 SAY CHR(27) + '!'+ 'Test'+ CHR(27) + '"'
```

Each character takes one space.
This prints the word "Test" in boldface and sets the type back to normal.

```
@ 5,50 SAY "Test"  <-------  Prints at column 46.
@ 6,50 SAY "Test"  <-------  Prints at column 50.
```

Access Disk

Because of a disk manufacturing error in which a unique access code was not recorded on System Disk #1 and System Disk #1 Back-up of dBASE III PLUS, Ashton-Tate implemented an exchange policy for the affected products. The serial numbers of the affected products are 2500001 through 2533960. Copies of dBASE III PLUS with other serial numbers are fully functional and are not affected.

The defect involves only local area network use. It does not affect functionality or data integrity of dBASE III PLUS when used in its stand-alone, single-user mode. The problem was created when a special access code that permits dBASE III PLUS to be networked with other copies of dBASE III PLUS was not placed on System Disk #1 and System Disk #1 Back-up. All other disks packaged with the product are not affected.

ASSIST

1. In any of the submenus of the Update, Position, Retrieve, or Organize menus that support the building of search conditions, when building a condition for a date type field, non-date entries are accepted as an argument for the CTOD() function without an error message. In the Developer's Release and earlier versions this would fail to locate any date field. But since the CTOD() function in dBASE III PLUS converts non-date entries to blank dates, searching for non-date entries on date fields with dBASE III PLUS will position the record pointer at the first blank date encountered. If there are no blank dates in the database file, dBASE III PLUS will not locate any records.

2. With versions of dBASE III greater than 1.1, it is not possible to build a search condition for blank spaces from ASSIST. The Developer's Release would return the user to ASSIST without comment when this was attempted. dBASE III PLUS will attempt to execute the command. However, when it constructs the command line, blank spaces are excluded. For example, requesting ASSIST to

   ```
   LIST FOR <Field> = " "
   ```

   actually executes:

   ```
   LIST FOR <Field> = ""
   ```

   Since all conditions are true when compared to a null string, this is identical to:

   ```
   LIST ALL
   ```

CASE

When using the CASE structure to branch control within your command files, there are two distinct behaviors if the either the DO CASE or ENDCASE statements are missing from the construct.

1. If a CASE statement is not preceded by a DO CASE statement, commands following the CASE statement and preceding the ENDCASE statement will not execute. In the following example, the LIST and WAIT commands do not execute but the DISPLAY MEMORY statement does.

   ```
   DO WHILE .T.
      USE Test
      CASE
         LIST <------|
         WAIT <------|__Commands do not execute.
      ENDCASE
      DISPLAY MEMORY <------- Command executes.
   ENDDO
   ```

2. A CASE structure without a terminating ENDCASE will terminate execution of a command file and return control to the dot prompt without an error message when the first true CASE statement executes.

CONFIG.DB

Using the keyword COMMAND in the CONFIG.DB or loading dBASE III PLUS with DBASE <Filename> alters the initial sign on. If this keyword is not included in the CONFIG.DB file and the command DBASE is used to call the program up, dBASE III PLUS will display the license agreement on the screen and immediately present the dot prompt. However, if COMMAND = <command> is included in CONFIG.DB or if dBASE III PLUS is invoked with a command filename on the DOS command line, the sign-on message will remain on the screen for approximately ten seconds or until **Return** is pressed before executing the command from CONFIG.DB or DOing the specified file.

DO WHILE

In versions of dBASE III earlier than the Developer's Release, the DO WHILE command will stop evaluating it's condition as soon as the expression is complete, ignoring characters beyond that point. This makes it possible for conditions with a seemingly invalid expression to be executed. For example,

```
DO WHILE EOF() = .T.
DO WHILE EOF() = .F. both equal DO WHILE EOF()
```

and,

```
DO WHILE .T. = EOF() equals DO WHILE .T.
DO WHILE .F. = EOF() equals DO WHILE .F.
```

In the Developer's Release and dBASE III PLUS the above conditions will return the error message, "Invalid operator."

INPUT

The INPUT command behaves differently in different versions of dBASE III. In versions 1.0 and 1.1, it accepts a carriage return as a valid response and does not create a variable. If one exists, it is left intact. Control is then returned to the calling program or the dot prompt. In the Developer's Release and dBASE III PLUS, however, INPUT does not accept a carriage return as a valid response. It scrolls the screen up one line and awaits valid input each time a carriage return is entered and control is not returned to the calling program or dot prompt until a valid input is accepted.

Interactive Mode

In dBASE III PLUS, you cannot continue command lines issued at the dot prompt with a semicolon. Previous versions supported this feature.

To continue a line beyond the width of the screen keep typing and the command will scroll left. In dBASE III PLUS, the command line is an editable 254 character, single line, horizontal scroll space. All keys that are valid in full-screen modes are valid at command line.

SET DATE FRENCH

The format of date variable display varies between versions 1.1 and the Developer's Release and dBASE III PLUS, when SET DATE is FRENCH. For example,

	Version 1.1	Developer's Release dBASE III PLUS
SET DATE FRENCH	DD.MM.YY	DD/MM/YY

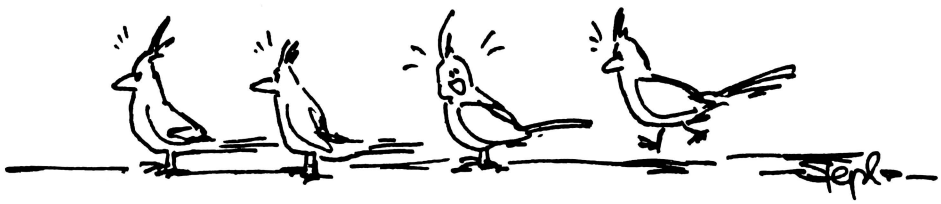
The Developer's Release and the dBASE III PLUS displays are identical to SET DATE BRITISH.

SET FIELDS TO

Issuing the SET FIELDS TO <Field list> command automatically SETs FIELDS ON, regardless of the FIELDS status prior to issuing the SET FIELDS TO command.

The *dBASE III PLUS Reference Manual* states on page 5-169 of the Using section, under the SET FIELDS TO command, "The list of fields is not active unless the SET FIELDS command is ON." This statement implies that the command SET FIELDS ON must be executed explicitly to activate the current field list after creating it. Anytime the command SET FIELDS is executed either to create a fields list or to add to one, FIELDS are SET ON.

SET ORDER TO



You must reposition the record pointer after executing SET ORDER TO in dBASE III PLUS. The easiest way to do this is to issue the command:

```
GO RECNO()
```

This will cause the record pointer to reposition itself to the current record position updating the internal record pointer. If, however, EOF() returns true (.T.), this won't work. In this case, use the following command:

```
GO IIF(EOF(), RECCOUNT(), RECNO())
```

SET SCOREBOARD/SET STATUS

The SET SCOREBOARD ON/OFF and SET STATUS ON/OFF commands will clear the entire screen when issued in dBASE III PLUS. In previous versions, SET SCOREBOARD ON/OFF would affect only the SCOREBOARD area of the screen.

SET VIEW TO

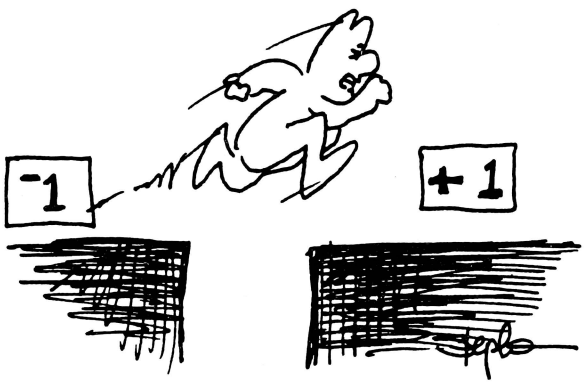
If any change has been made to any of the component parts of a view, invoking that view file with SET VIEW TO will return an error message. The specific message will depend on the error in the view file.

For example, if the name of field in the fields of the view has been changed by MODIFYing the STRUCTURE of one of the constituent database files, issuing the SET VIEW TO command will return "Variable not found." Once this condition has occurred, there is no way to open the view file and modify it, as MODIFY VIEW will return the same error message. The file must be deleted and re-created, or the environment must be made consistent with the view file once more.

ANOMALIES

dBASE II

SKIP



If a space precedes the expression argument of the SKIP command in dBASE II, version 2.41 and 2.43*, the result is a syntax error.

For example, the command,

```
SKIP - 3
       ^-- a space here
```

or

```
SKIP+ 4
      ^-- a space here
```

will result in a syntax error.

dBASE III PLUS

@...SAY/GET

Using @...SAY/GET to display information or scrolling text to line 0 with SET SCOREBOARD ON and SET STATUS OFF, will cause the information to remain on line 0 and not scroll off the top of the screen as new information is displayed. For example, the following block of code will cause this anomaly.

```
SET SCOREBOARD ON
SET STATUS OFF
item = "123456789"
@ 0,15 SAY "This is a test."
@ 5,10 SAY "What is this?" GET item
READ
```

In order to erase this line, you must use @ 0,0, CLEAR, SET SCOREBOARD ON/OFF, or SET STATUS ON/OFF.

To prevent conflict, avoid the use of line 0 with SET SCOREBOARD ON. Be aware, however, when using CREATE SCREEN that the first line on the blackboard is line 0. Additionally, IMPORT also writes to line 0 when creating its format file.

CREATE/MODIFY SCREEN

When you use the <NewField> option of the Contents submenu of the Modify menu to add a new field to the current database file, it is automatically created as a character field with a length of one and all the records of the previous structure are APPENDed into the new one at that point. To create a new field that is not character type or has a length of greater than one, you must modify the attributes of the new field after it has been created.

Be aware that if you are MODIFYing the STRUCTURE of a database file with a large number of records, each time you add or change an attribute of a field, the entire database file will be APPENDed into the new structure. Multiple changes necessitates multiple APPENDs. If you are anticipating making extensive changes to your database file structure from within CREATE/MODIFY SCREEN, use the following method:

```
COPY STRUCTURE TO <a temporary database file>
USE <temporary database file>
CREATE/MODIFY SCREEN <your screen>
```

CREATE/MODIFY VIEW

1. You cannot permanently delete a filter formula in MODIFY VIEW. It appears to allow you to delete the filter and then SAVE the VIEW, but the next time MODIFY VIEW is invoked with the same (.VUE) file, the original filter formula will still be there. The only way to delete the filter formula is to SET VIEW TO the (.VUE) file, SET FILTER TO, and CREATE VIEW FROM ENVIRONMENT.
2. You cannot include a file from any subdirectory other than the current directory using CREATE VIEW FROM ENVIRONMENT. The default alias name is included in the view file, returning "Alias already in use" when the file is accessed with the SET VIEW TO command.

FOUND()

If you have SET DELETED ON and perform FIND or SEEK on a key with multiple occurrences, where the first occurrence is deleted, the FOUND() function will return a .F., even though a matching record has been found.

The same test can be made with the EOF() function. After the SEEK or FIND command, EOF() will return a true value (.T.), if there is not a match.

INDEX

The SET CENTURY ON command and the DTOC() function are not compatible for the purpose of creating an index file. INDEXing ON the DTOC() of a date field with SET CENTURY ON will appear to index the file correctly. The file may be accessed, the FIND or SEEK commands may be used, but any attempt to BROWSE, EDIT, or GOTO a record will produce the error message, "Record not in index."

READKEY()

If a full-screen operation is terminated by **Ctrl-W** or **Ctrl-End** in dBASE III PLUS, the READKEY() function will always return the value 270, regardless of whether the contents have been altered or not. The READKEY() function should return a value of 14 if any of the pending GETs were not changed and **Ctrl-W** or **Ctrl-End** was pressed.

The Developer's Release, READKEY() returns the expected value 14.

MULTIMATE TECHNICAL SUPPORT

As you may know, Ashton-Tate recently acquired MultiMate International who manufactures word processing software and add-on products. One of the results of the Ashton-Tate/MultiMate merge is that ***TechNotes*** will be offering technical support information on MultiMate products, beginning with this issue.

This month presents information about MultiMate's newest version of Advantage which offers column features and a thesaurus. There is also information on how and when to contact MultiMate technical support and a "checklist" of other resources you can use to get answers to your technical questions on MultiMate products.

We hope you can make use of the MultiMate technical information found in ***TechNotes***. Please continue to contact MultiMate for technical support on MultiMate products and Ashton-Tate for dBASE and Framework.

(Liz Burke, MultiMate Technical Support, contributed this information for publication in ***TechNotes***. Some of the following material originally appeared in MultiMate's *On-Line* newsletter.)

MultiMate Advantage Update

A new version of the MultiMate Advantage Professional Word Processor includes several of the most user-requested word processing enhancements.

MultiMate Advantage Version 3.60 provides users with two kinds of column capability, a built-in thesaurus, a typewriter mode, and a utility for custom dictionary editing, among other new features.

Column Capability

MultiMate Advantage Version 3.60 supports as many as eight bound columns (information is horizontally associated) on a page. Users are able to set the column width and tabs for each column, as well as the spacing between columns.

The new version also supports snaked text, which runs continuously from the top to the bottom of a column and then to the top of the next column, as in newspaper columns. All editing functions are supported in the snake column mode and text is reformatted automatically.

Thesaurus

A 40,000-word Merriam-Webster thesaurus provides definitions as well as alternative words and phrases of up to 50 words. The thesaurus was developed by Proximity Devices Incorporated of Fort Lauderdale, Florida.

Typewriter Mode

A typewriter mode allows the user to create and print text by character or by line from the Print Control Utilities menu. This feature permits a personal computer to function like an electronic typewriter with direct keyboard-to-printer operation, making small jobs, such as typing a single envelope, much easier. Version 3.60 also includes a "hot" print feature that allows the user to print one page of a document directly from the document screen.

Custom Dictionary Editing

A new utility in Version 3.60 allows for the creating, editing, and deleting of words in a custom dictionary. Words can also be deleted from a custom dictionary during spell editing.

Other new MultiMate 3.60 features include:

- Keyboard merge: allows the user to insert variable information in a document without having to create a secondary document. For example, if three letters must be printed, the keyboard merge allows the user to create the letter and then insert the names and addresses just before printing.
- A 110,000-word Merriam-Webster speller/dictionary: includes legal and medical terminology.
- Search capability for footnote symbols, section numbers, and double underscoring.
- Additional proportional spacing support for printers, including the IBM Quietwriter.
- Document conversion utilities for files created on the WANG PC, and Lotus 1-2-3 .WKS files. (Other software product file formats available on request.)
- Increased cursor response with adjustable rate options.

System Requirements

MultiMate Advantage Version 3.60 is designed for the IBM PC, XT, AT, and Compaq, plus all IBM-compatible computers. It requires a minimum of 320K RAM for use with DOS 2.0 or 2.10, and a minimum of 320K free RAM for DOS 3.0 and higher, two double-sided disk drives or a hard disk.

MultiMate Advantage Version 3.60 is available to registered MultiMate users as a product update or upgrade. Version 3.60 began shipping March 1, 1986.

MultiMate GraphLink

Registered MultiMate users may also order the new MultiMate Text and Graphics Integrator. GraphLink allows any image produced by most graphics-generating software for the IBM PC to be captured and printed within a MultiMate Advantage document. Included are spreadsheets, pie charts, freehand illustrations, and signatures.

System requirements for GraphLink are an IBM or compatible color/printer board, DOS 2.0 or higher, and 85K RAM in addition to the memory required for the applications software.

NOTE: Only registered MultiMate users are eligible to update.

Where to Look When You Have Trouble

We have found that the most common MultiMate questions can be answered by checking these resources in the following order:

Where to look for answers to MultiMate technical questions:

1. MultiMate Beginner's Guide
2. MultiMate Training Lessons
3. MultiMate Tutorial
4. MultiMate Reference Manual
5. MultiMate *On-line* Newsletter
6. MultiMate library on Compuserve
7. Dealer from whom you bought the software

When to call MultiMate Technical Support:

1. When you have trouble getting started with MultiMate
2. When you have problems accomplishing certain tasks with MultiMate
3. When you get an undefined error message in MultiMate (Check your error message section in the reference manual.)
4. When you need to know how to integrate MultiMate with another software program
5. When you believe you have a defective original diskette

About MultiMate Technical Support

Supported Products

The MultiMate Technical Support Department is dedicated to handling technical support questions from both beginning and advanced users of the MultiMate product line. Our technicians answer questions on the following MultiMate products:

- MultiMate Advantage
- MultiMate 3.3 Series
- On-File
- GraphLink
- Just Write
- LAN versions of MultiMate
- MultiMate Business Advantage Communications board

Calling MultiMate Technical Support

MultiMate International's Technical Support department is open from 9 a.m.—7 p.m. EST (except holidays) to answer your questions about using MultiMate products. For the fastest answer to your question, dial directly to Tech Support at (203) 247-3445. Please do not call MultiMate's main switchboard for technical support. If a technician is not immediately available when you call, you may leave a message to have your call returned. Calls are returned as soon as possible, usually within two hours. Be sure to have your MultiMate serial number on hand when you call. Registered MultiMate users are given a higher priority on returned calls.

Customer Service or Technical Support?

The differences between MulitMate's Customer Service and Technical Support departments are not always apparent to end users. If you're uncertain about which department to call about a particular problem, or if you call the wrong department, you can end up wasting a great deal of time waiting on hold or talking to the wrong department.

Each department has specific functions to help you solve particular problems:

Customer/Update Services (203) 522-2116

- Registration verification
- End-user product updating information
- Lost serial numbers
- Missing pages from the user's manual
- Change of address for MultiMate *On-Line*
- Replacing defective diskettes (only after Technical Support has verified the disks as defective)

Technical Support (203) 247-3445

Your MultiMate serial number is required to receive help for the following:

- MultiMate technical questions
- Printer support questions
- Requests for Printer Action Tables
- MultiMate applications questions
- Hardware questions relating to MultiMate
- Clarification of MultiMate functions (e.g., Merge, Key Procedures)

You can further avoid unnecessary delays by doing the following before calling Tech Support:

1. Make sure your PC, printer, cables, peripherals, and MultiMate Word Processor are set up and used exactly as described in the product manuals.
2. Consult the Reference Section of your MultiMate user's manual to find an answer to your question or problem.
3. When it is necessary to call Tech Support, be in front of your computer with MultiMate loaded so that you can be ready to try the solutions provided by the technician while still on the phone.
4. Have all applicable information ready when you call, including MultiMate version number, serial number, type of printer, type of PC, amount of memory, what you were attempting to accomplish, and what error/machine message you received.

LETTERS

Importing Lotus 1-2-3 Release 2 Files into Framework II

Dear *TechNotes*:

I am a user of both Framework II and Release 2 of Lotus 1-2-3, and I am attempting to transfer Lotus 1-2-3 files into Framework II. Framework II has a utility to import Lotus 1-2-3 version 1 files. Fortunately, Lotus has a utility to translate version 2 files into version 1 files. However, due to the string function handling being different between version 2 and version 1, some formulas end up as text. Is there an easy way in Framework II to convert text in spreadsheets into formulas?

R.G.

Dear R.G.:

The Framework II Library cabinet contains two macros that will allow you to "cut" the text from the contents area of the cell and "paste" it into the formula area. The {Alt-F7} macro will copy selected text to a buffer, and the {Alt-F8} will copy the contents of the buffer to the desired location. These are more commonly known as "cut and paste" macros. In your example, follow the steps below:

1. Highlight the desired text using **EXTEND SELECT (F6)**.
2. Once selected, press **Alt-F7**.
3. Press **Del** key to remove the contents of that cell.
4. Press **EDIT FORMULA (F2)** to move behind the cell.
5. Press **Alt-F8** to copy the text into the formula area.

Sending Line Feeds to Your Printer

Dear *TechNotes*:

I am using dBASE II. I am trying to send to my printer a series of line feeds without the carriage return. I tried using a ? on a line by itself, but this also sends a carriage return. I also tried using ?? on a line by itself, but this does not work. How can I get a series of line feeds without the carriage returns?

L.W.

Dear L.W.:

A ? command will send a carriage return and line feed. A ?? command will not send the carriage return and line feed. The CHR() codes for carriage return and line feed are CHR(13) and CHR(10), respectively. Therefore, to send just a line feed to the printer, issue the command:

```
?? CHR(10)
```

Creating Index Files in a Subdirectory

Dear *TechNotes*:

I have dBASE III set up on my hard disk under a subdirectory called DBASE. Since I have several programs and database files, I have placed these files in sub-subdirectories below the DBASE subdirectory. One of my sub-subdirectories, named CUSTOMER, contains the database file Names.DBF. When I enter dBASE III, I type:

```
SET PATH TO C:\DBASE\CUSTOMER
```

This will allow me to USE Names. However, when I issue the command

```
INDEX ON Name TO Name
```

the index file is created in the DBASE subdirectory and not in the CUSTOMER sub-subdirectory. How can I create an index file that will follow the PATH?

C.C.

Dear C.C.:

The SET PATH TO command does not work the same way as SET DEFAULT TO command. The SET PATH TO command means that if you try to USE a database file or DO a program, dBASE III will first check the current directory. If dBASE III cannot find the file in the current directory, it will then search the PATH.

There are two ways to create an index file in a particular sub-subdirectory. First, you can change directories by issuing the RUN command (assuming you have more than 256K). After entering dBASE III, type:

```
RUN CD\DBASE\CUSTOMER
```

This makes CUSTOMER the current sub-subdirectory and returns you to the "dot" prompt. At this point, you can issue the commands:

```
USE Names
INDEX ON Name TO Name
```

and Name.NDX will be created in the CUSTOMER subdirectory.

The second alternative is to specify the target directory's path as the filename when creating the file. For example,

```
INDEX ON Name TO \DBASE\CUSTOMER\Name
                 ↑
                 └──────────── Path name
```

Using IIF() for Conditional Indexing in dBASE III PLUS

Dear ***TechNotes***:

I am using dBASE III PLUS. One database file, Company.DBF, has over 15,000 records. Contained in Company.DBF are two fields, Name1 and Name2. Name1 contains contact names, and Name2 contains the names of salespeople. With some companies, a salesperson does not exist, so Name2 is left blank. With other companies, if the contact is the salesperson, that name is entered once into Name2, and Name1 is left blank. This leaves us in a strange predicament. At times, we need a report listed alphabetically by salesperson (Name2). The problem lies when Name2 is blank. At this point, we would like to list the contact (Name1), and still keep the list in alphabetical order. Since the file is INDEXed ON Name2 + Name1, the blank salespeople (Name2) are listed first with an alphabetical listing of contacts (Name1) and then followed by another alphabetical listing of salespeople. Is it possible to get one alphabetical listing as described above?

R.M.

Dear R.M.:

This is a very good question. It can be accomplished in the Developer's Release and dBASE III PLUS. The IIF() function makes it possible to put IF...ELSE...ENDIF commands onto one line. For example:

```
IF mamt >= 0
message = "POSITIVE"
ELSE
message = "NEGATIVE"
ENDIF
```

can now be issued as:

```
message = IIF(mamt >= 0,"POSITIVE","NEGATIVE")
```

The IIF() function contains three parameters:

1. The condition
2. The true portion of the condition
3. The false portion of the condition

Using your example, we can set up an index key that contains Name2, and when Name2 is blank, we use Name1. By using the above syntax, issue the following commands:

```
USE Company
INDEX ON IIF(Name2 = " ",Name1,Name2) TO Compname
LIST IIF(Name2 = " ",Name1,Name2)
```

and your alphabetical listing will appear.

We can carry this algorithm over to the REPORT FORM, too. Enter for the first field contents:

```
IIF(Name2 = " ",Name1,Name2)
```

If we want to know if the person is a contact or salesperson, enter for the second field contents:

```
IIF(Name2 = " ","Contact","Salesperson")
```

(For more information, read Christopher White's article, "The IIF() Function," on page 8 of the March 1986 issue of ***TechNotes***.)

Direct your technical correspondence to Ashton-Tate Software Support, 20101 Hamilton Avenue, Torrance, CA 90502-1319. Be sure to include information on your hardware configuration when appropriate, such as the type of printer, hardware, and version of operating system you are using.

Software Support Technician
Take the Chance to Advance

If you are seeking greater career challenges in a progressive thought-provoking atmosphere of software innovation, consider the excellent opportunities available at ASHTON-TATE.

We are currently seeking a Software Support Technician with working knowledge of dBASE II and dBASE III as well as familiarity with MS-DOS and CP/M operating systems. This assignment will involve going on-line and interfacing with ASHTON-TATE developers, dealers and end-users. You should have proven problem-solving skills and the ability to write dBASE programs.

Find out how you can enjoy professional challenges and rewards. Send your resume to: **Tony Faison, ASHTON-TATE, Dept. TN-0100, 20101 Hamilton, Torrance, CA 90502.** An equal opportunity employer.

Update Policies

dBASE II

The latest version of dBASE II, version 2.43*, is now available for update to all registered users of dBASE II, subject to format availability.

In order to update your current version of dBASE II, you must have your original software registration card on file with Ashton-Tate Customer Service. If you are not a registered user, you may request that a registration card be sent to you.

Prices for this update are outlined below, depending on which version of dBASE II you are currently using. (Funds must be in US dollars drawn on a US bank.)

Version 2.3
$75.00 within US — UPS shipping
$90.00 within US — DHL (3 day)
$100.00 in Canada - DHL (only)

Version 2.4
$50.00 within US — UPS shipping
$60.00 within US — DHL (3 day)
$75.00 in Canada — DHL (only)

Version 2.41 (purchased before 12/1/84)
$25.00 within US — UPS shipping
$35.00 within US — DHL (3 day)
$40.00 in Canada — DHL (only)

If your product was purchased after April 1, 1985, the 2.43* update is at no charge. Please include proof of purchase with requests for free updates. Direct all correspondence to:

Ashton-Tate Customer Service
dBASE II Update
20101 Hamilton Ave.
Torrance, CA 90502-1319
(213) 329-1664

dBASE III PLUS

Registered users of dBASE III version 1.0 or 1.1 will be notified by mail regarding the upgrade to dBASE III Plus. **Unregistered** users of dBASE III version 1.0 or 1.1 who wish to upgrade to dBASE III Plus are required to send in their registration card and contact: (213) 329-1664, or (213) 329-9989.

Framework II

Registered users of Framework version 1.0 or 1.1 will be notified by mail regarding the upgrade to Framework II. **Unregistered** users of Framework version 1.0 or 1.1 who wish to upgrade to Framework II are required to send in their registration card and contact: 1(800)437-4329.

Disk Replacement Policy

In the event that you find a defect in the programming, materials, or workmanship of an Ashton-Tate system disk within three months of purchase, we will replace it free of charge. Call Software Support first to verify the defect, (213) 329-0086, between the hours of 6:00 am and 5:00 PM, PST.

If you damage your system disk anytime within three years of purchase (and after the 90-day warranty), there is no need to call Software Support. Our Customer Service department will replace it for $20.00 (per disk), payable by check, money order, or VISA/MasterCard (U.S. funds drawn on U.S. bank). Just complete the replacement form below and send it to Customer Service at the address below, along with the defective disk and your payment. (Please enclose a dated proof-of-purchase.)

Ashton-Tate Customer Service
Disk Replacement Form
20101 Hamilton Ave.
Torrance, CA 90502-1319

PLEASE BE ADVISED: You **must** be **registered** with Ashton-Tate in order to receive a replacement disk. If you have lost your registration card, or feel that you may not be on file, please call Customer Service at (213) 320-1664, Monday through Friday, 8:00 am to 5:00 PM, PST.

DISK REPLACEMENT FORM

Name _____

Company _____ Dept./Div. _____

Address _____ Suite/Room _____

City _____ State _____ Zip _____

Country _____ Phone ( )_____-_____

* System Disk Serial Number (REQUIRED) _____

Ashton-Tate Product Used _____

Computer Brand and Model Used _____

AMOUNT ENCLOSED: $_____

NOTE: You must either insure any defective system disk being returned or assume the risk of loss or damage in transit.

* If your disk indicates your serial number as being "ON FILE," refer to your original system disk or system backup disk for your serial number.

TechNotes Reader Survey

In our effort to make ***TechNotes*** an even more valuable resource to you, we want your thoughts about it. Please take a minute to fill in the following questions, and return it to us in the postage-paid return envelope. Share with us your comments so that we may keep ***TechNotes*** in tune with your needs. Thank you.

1. Which of the following statements best describes your *overall feeling* about ***TechNotes***?

   Excellent, almost always has useful information for me. ___
   Good, frequently has useful information for me. ___
   Average, occasionally has useful information for me. ___
   Fair, seldom has useful information for me. ___
   Poor, almost never has useful information for me. ___

2. Which Ashton-Tate product(s) do you use one or more times a week? (Check all that apply.) ___None

   ___ dBASE II ___ dBASE III ___ Developer's Release
   ___ dBASE III PLUS ___ Framework ___ Framework II

3. How long have you used Ashton-Tate products? ___ yrs ___ mos.

4. How long have you subscribed to ***TechNotes***? ___ yrs ___ mos.

5. Do you keep your back issues of ***TechNotes***? ___ Yes ___ No

6. How many people, *besides you*, read your copy of ***TechNotes***?___

7. Please rank from 1 to 9 the following types of information in order of their *usefulness to you*. (1 is most useful, 9 is least useful.)

   ___ Technical Articles ___ Anomalies
   ___ Application Articles ___ Sample Programs
   ___ Letters from users ___ Update Policies
   ___ Usage Tips ___ ***TechNotes*** Index
   ___ Your Choice _____

8. When you have a technical question or problem related to Ashton-Tate products, what do you do *first*?

   ___ Call Software Support
   ___ Read ***TechNotes***
   ___ Connect to On-Line Services (CompuServe, The Source, etc.)
   ___ Read the documentation
   ___ Call a knowledgeable friend
   ___ Other _____

9. Which MultiMate® product(s) do you use? (Check all that apply.)

   ___ None
   ___ MultiMate Advantage™ ___ On-File™
   ___ MultiMate 3.3 Series ___ GraphLink™
   ___ MultiMate Executive™ ___ Business Advantage™ Keyboard
   ___ Just Write™ ___ Business Advantage Communications Board

10. Would you like to see topics about MultiMate software covered in ***TechNotes***? ___ Yes ___ No

    If yes, what topics?

    1._____

    2._____

11. Would you like to see *advertisements* in ***TechNotes*** on products and services related to Ashton-Tate products? ___ Yes ___ No

    If yes, what products or services would you like to see advertised?

    1._____

    2._____

    If no, why not?_____

    _____

12. Would you like to see *case studies* in ***TechNotes*** on how Ashton-Tate software is being used to solve business problems? ___ Yes ___ No

    If yes, do you have a suggestion for a case study?

    _____

13. Which of the following would you like to see expanded in coverage and which reduced in coverage in ***TechNotes***?

	Expand	**Reduce**	**Stay as is**
dBASE II	___	___	___
dBASE III	___	___	___
Developer's Release	___	___	___
dBASE III PLUS	___	___	___
Framework 1.x	___	___	___
Framework II	___	___	___
Usage Tips	___	___	___
Anomalies	___	___	___
Letters	___	___	___
Policies	___	___	___
MultiMate	___	___	___

14. What, if any, additional topics would you like to have included in ***TechNotes***, in order of your preference?

    1._____
    2._____
    3._____

15. What is the single thing that you *like most* about ***TechNotes***?

    _____

16. What is the single thing that you *like least* about ***TechNotes***?

    _____

17. What other comments would you like to make regarding any topic?

    _____
    _____
    _____

Thank you for your valuable input.

SUBSCRIBE TODAY TO *TechNotes*!

Don't miss a single issue of Ashton-Tate's hottest source of technical know-how! Each monthly issue of ***TechNotes*** provides you with the most current, accurate technical information available from the Software Support Center on dBASE II, dBASE III PLUS, and Framework II.

Call (619) 747-1666 to order your one year subscription (12 issues) by phone, or fill out the form below and return it with payment to:

TechNotes Subscription Service
P.O. Box 3729
Escondido, CA 92025-0929

— — — — — — — — — — — — — **(CUT HERE)** — — — — — — — — — — — — —

Make checks payable to ASHTON-TATE.

Amount Enclosed:

__ $50.00

__ $65.00 (Extra Postage Canada and Mexico)

__ $75.00 (Extra Postage other foreign countries)

__ Bill Me

Form of Payment: (check one)

__ Check (**U.S. dollars drawn on U.S. bank**)

__ VISA Number _____

__ MasterCard Number _____

Expiration Date _____

Signature _____

Check One: __ New Subscription __ Renewal

Name _____

Company _____

Address _____

City _____ **State/Province** _____

Zip/Postal Code _____ **Country** _____

Which product(s) do you use? __ **dBASE II** (D2)
__ **dBASE III** (D3)
__ **Framework** (FW)

Please allow 30 to 60 days for delivery of your first issue.

1000

TechNotes

BACK ISSUES ORDER FORM

Maybe your subscription to ***TechNotes*** just began and you want to have every issue. Maybe you don't subscribe, but want to receive last month's issue. Maybe you want to refer to an article in a past issue to help you design that special application.

Whatever the reason, order Back Issues! Add to your library of dBASE II, dBASE III, and Framework technical reference material. To order your Back Issue(s), fill out this form, enclose your payment, and return to:

TechNotes Back Issues
Ashton-Tate Software Support Center
20101 Hamilton Ave.
Torrance, CA 90502-1319

Name _____

Company _____

Address _____

City _____ State/Province _____ Zip Code _____

Country _____ Phone Number (_____) _____-_____

All 1985 Back Issues are $10.00 each, when you order them separately. If you purchase all 1985 issues, the price is $50.00. All 1986 Back Issues are $7.00 each. All prices include postage and tax.

The following issues are currently available. Check those you wish to order:

( ) FEBRUARY 1985
( ) MARCH 1985
( ) APRIL 1985
( ) MAY 1985
( ) JUNE 1985
( ) JULY 1985
( ) AUGUST 1985
( ) SEPTEMBER 1985
( ) OCTOBER 1985
( ) NOVEMBER 1985
( ) DECEMBER 1985 (includes 1985 Annual Disk and 1985 Cumulative Index)
( ) JANUARY 1986
( ) FEBRUARY 1986
( ) MARCH 1986

$_____ TOTAL AMOUNT ENCLOSED

( ) VISA ( ) MasterCard # _____

Expiration Date _____ Signature _____

ALL ORDERS MUST BE PREPAID. MAKE CHECKS PAYABLE TO ASHTON-TATE (IN U.S. DOLLARS DRAWN ON U.S. BANK). PLEASE ALLOW SIX (6) WEEKS FOR DELIVERY OF YOUR ORDER.

NOTE: Information contained in ***TechNotes*** is current as of the issue in which it appears. Therefore, some information found in the Back Issues may no longer be valid (such as Update Policies, etc.).

READER SUBMISSION FORM

If you have a programming tip, a handy formula, or a short utility, and you are interested in sharing it with other dBASE II, dBASE III, and Framework enthusiasts in a future issue of ***TechNotes***, please read the provisions below. After reading these guidelines, fill out this form and send it along with your submission(s) to the following address. Submissions must be made on disk (IBM or compatible format) in standard ASCII format or in Framework (.FW) format.

TechNotes Submissions
Software Support Center
Ashton-Tate
20101 Hamilton Ave.
Torrance, CA 90502-1319

NAME _____ DATE _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PHONE (During normal business hours) ( ) _____ - _____

_____

(SIGN HERE TO ACKNOWLEDGE YOUR UNDERSTANDING OF THE PROVISIONS BELOW)

To establish a clear understanding of the rights and usage of submissions, you are asked to first represent and warrant that:

1. You are the author of the material: that it is original and has not been heretofore published; that it does not infringe upon any copyright, proprietary right or any other right whatsoever; that it contains no matter that is contrary to law; and that you are the sole owner of the rights herein conveyed to us, and that such rights are free and clear.
2. We shall have the right to use your name, biography or other information pertinent to your submission.
3. We shall have the right, at our discretion, to edit the material to suit the requirements of our publication.
4. Suggestions, comments, and programming materials submitted to ***TechNotes*** are not for reply. Materials submitted for publication are considered contributions to the "pool" of information found in ***TechNotes*** for the advantage of our readers. Although, we do not pay or in any other way compensate you for your contribution, your name shall appear as author or contributor.

Thank you for reading ***TechNotes***. If you do not have a submission for us, write us anyway and let us know what you think about our publication. We certainly appreciate any feedback you may have.

CUMULATIVE INDEX

January 1986 through April 1986

Articles

Subject	Product	Issue	Page	Title and Description
256K ENVIRONMENT	**dBASE III PLUS**	Apr	20	**Operating dBASE III PLUS in a 256K Environment** *Tips for operation in this environment*
@...SAY	**dBASE III PLUS**	Mar	11	**The IIF() Function** *Using this new function for conditional displaying*
@isabend	**Framework II**	Feb	38	**The Protean Powers of @isabend** *Explanation of this powerful function which tests for error conditions*
@panel1/@panel2	**Framework II**	Jan	14	**Overview of New Framework II Features** *Explanation of these new functions*
ABBREVIATIONS	**Framework II**	Feb	24	**Library Macros and Abbreviations** *How to use, edit, and create abbreviations in the Library*
ARRAYS	**dBASE III PLUS**	Apr	9	**Arrays in dBASE TOOLS FOR C** *Interfacing with C from dBASE III PLUS*
ASSEMBLY LANGUAGE	**dBASE II**	Feb	13	**Interfacing with the Machine** *How to use the assembly language interface with dBASE II*
BOXES	**Framework II**	Apr	39	**How to Use Boxes in Word Frames** *A creative way to work with documents*
CATALOG	**dBASE III PLUS**	Apr	26	**An Example of Catalog Management** *Programs for working with the Catalog*
DATA	**dBASE III**	Feb	31	**Data Encryption Strategies** *Programs for data security in dBASE III*
	dBASE III PLUS	Feb	11	**Toggling Between Table and Form Views** *Program to toggle between table view and form views in dBASE III Plus*
DOS	**Other**	Mar	29	**Generating Menus in PC/MS-DOS** *How to use batch files to create menus in DOS*
END-OF-FRAME	**Framework II**	Feb	40	**The Protean Powers of @isabend** *Using @isabend to test for boundaries of frames*
ERROR CHECKING	**Framework II**	Feb	39	**The Protean Powers of @isabend** *Using @isabend to check for error conditions*
FINANCIAL MODELS	**Framework II**	Feb	34	**Data Tables in Framework II** *Creating data tables for "what-if" modeling*
HISTORY	**dBASE III PLUS**	Feb	7	**Using History in dBASE III PLUS** *Using HISTORY as an algorithm-building tool*
IIF()	**dBASE III PLUS**	Mar	8	**The IIF() function** *Examination of this new dBASE III PLUS function*
IMPORT	**Framework II**	Feb	28	**Framework II for Framework 1.x users** *Importing Framework 1.x files into Framework II*
		Mar	36	**Converting PFS files to Framework II** *Importing files from pfs:file to Framework II*

Articles

Subject	Product	Issue	Page	Title and Description
INDEX	**dBASE III PLUS**	Mar	10	**The IIF() Function** *Using this function to INDEX on logical fields and conditional keys*
INSTALLATION	**Framework II**	Feb	28	**Framework II for Framework 1.x users** *Framework II installation procedures for Framework 1.x users*
INTERVIEW	**Framework**	Feb	3	**Technotes Interview with Robert Carr** *Part 2, Framework creator discusses possible plans for the future*
		Jan	4	**Interview with Robert Carr** *Part 1, the creater of Framework discusses its development*
LABEL FORM	**dBASE III PLUS**	Mar	9	**The IIF() Function** *Using this new function to format output*
LIBRARY	**Framework II**	Apr	34	**A Guided Tour of the Library Cabinet** *A tour of this new Framework II feature*
		Feb	24	**Library Macros and Abbreviations** *Introduction to the Library cabinet*
		Jan	32	**Creating String Functions***Adding functions and macros to Framework II Library*
LOGICAL FIELDS	**dBASE III PLUS**	Mar	10	**The IIF() Function** *Using IIF() to INDEX on logical data*
MACLIB	**Framework II**	Feb	28	**Framework II for Framework 1.x users** *Transporting MacLib into Framework II*
MACROS	**Framework II**	Feb	24	**Library Macros and Abbreviations** *How to use, edit, and create macros in the Library*
MAILING LABELS	**Framework II**	Mar	14	**Automating Mailing Labels** *FRED program to ease the generation of mailing labels*
MENUS	**dBASE III PLUS**	Mar	12	**SET COLOR from a Menu** *A program for defining color from a pull-down menu*
	Framework II	Feb	21	**Overview of new Framework II features** *New options in the Locate, Frames, Graph, and Print pull-down menus*
		Jan	14	**Overview of New Framework II Features** *New options in the Disk, Create, and Edit pull-down menus*
	Other	Mar	29	**Generating Menus in PC/MS-DOS** *How to use batch files to create DOS menus*
MODIFY STRUCTURE	**dBASE II**	Jan	10	**An Alernative to MODIFY STRUCTURE** *A dBASE II program to emulate this command*
NEW FEATURES	**dBASE III PLUS**	Apr	26	**An Example of Catalog Management** *Working with the Catalog*
		Feb	7	**Using HISTORY in dBASE III PLUS** *Using this new feature as an algorithm-building tool*
		Mar	4	**Understanding Views** *Discussion of views as they enhance relational operations*
		Mar	8	**The IIF() Function** *Application uses of this new function*
		Mar	12	**SETCOLOR from a Menu** *A program for defining color from a pull-down menu*

Articles

Subject	Product	Issue	Page	Title and Description
NEW FEATURES	**Framework II**	Apr	3	**Telecommunications Primer** *Framework II telecommunications—Part one: setting up*
		Apr	34	**A Guided Tour of the Library Cabinet** *Macros, Abbreviations, Templates, User-Defined Functions, and more*
		Feb	21	**Overview of New Framework II features** *New Locate, Frames, Graph, and Print pull-down menu options*
		Feb	24	**Library Macros and Abbreviations** *Introduction to the Library Cabinet*
		Feb	28	**Framework II for Framework 1.x Users** *Explanation of installation and configuration for users who upgraded*
		Feb	38	**The Protean Powers of @isabend** *Using this new functions to test for error checking*
		Jan	14	**Overview of New Framework II features** *@panel1, @panel2; New Disk, Create, and Edit pull-down menu options*
		Jan	32	**Creating String Functions** *Adding custom functions and macros to the Library*
		Mar	27	**Using Rundbase on a Hard Disk** *Program to run a hard-disk based dBASE III from Framework II*
PFS:FILE	**Framework II**	Mar	36	**Converting PFS Files to Framework II** *Program to transfer the files*
PRIMITIVES	**dBASE III PLUS**	Apr	9	**Arrays in dBASE TOOLS FOR C** *Interfacing with the C language from dBASE III PLUS*
PRINTING	**dBASE III**	Mar	21	**Printing Program Listings** *Program to print dBASE III program files*
RANDOM NUMBERS	**dBASE**	Jan	17	**Random Numbers Generation for dBASE** *Random number generation in dBASE II and dBASE III*
REPORT FORM	**dBASE III PLUS**	Mar	8	**The IIF() Function** *Using this new function to format REPORT FORM output*
REPORTING	**dBASE III PLUS**	Apr	2	**Listing a Database File in a Verticle Form** *An alternative way to LIST a database file*
ROBERT CARR	**Framework**	Feb	3	**Technotes Interview with Robert Carr** *Interview with the creator of Framework, part 2*
		Jan	4	**Interview with Robert Carr** *Interview with the creator of Framework, part 1*
RUNDBASE	**Framework II**	Mar	27	**Using Rundbase on a Hard Disk** *Program to run a hard-disk based dBASE III from Framework II*
SET COLOR	**dBASE III PLUS**	Mar	12	**SET COLOR from a Menu** *A program which generates a menu for color definition*
SETUP	**Framework II**	Feb	28	**Framework II for Framework 1.x users** *Replaces installation and configuration programs in Framework*
SPREADSHEET	**Framework II**	Feb	34	**Data Tables in Framework II** *Creating data tables for "what-if" modeling*
		Mar	25	**Exchanging Numbers and Text in a Cell** *Useful macros for Framework II spreadsheets*
		Jan	7	**Framework Interview with Robert Carr** *Evolvement of spreadsheets in Framework*

Articles

Subject	Product	Issue	Page	Title and Description
TABS	**Framework II**	Jan	34	**Removing Tabs from a Word Frame** *Program used on text files before or after transmitting via telecom.*
		Mar	18	**Variable Tab Stops** *Creating variable tab stops in Framework II*
TELECOMMUNICATIONS	**Framework II**	Apr	3	**Telecommunications Primer** *An introduction to hardware and software setup*
		Jan	34	**Converting PFS files to Framework II** *Program used on text files before of after transmitting*
		Jan	34	**Removing Tabs from a Word Frame** *Program used on text files before or after transmitting*
VIEWS	**dBASE III PLUS**	Mar	4	**Understanding Views** *Discussion of views as they enhance relational operations*
WORD FRAMES	**Framework II**	Apr	39	**How to Use Boxes in Word Frames** *A creative way to work with documents*
dBASE TOOLS FOR C	**dBASE III PLUS**	Apr	9	**Arrays in dBASE TOOLS FOR C** *Interfacing with the C language from dBASE III PLUS*

Usage Tips

Product	Entry	Issue	Page	Description
dBASE II	@...SAY	Jan	37	*Patches to access line 24 with @...SAY in versions 2.4 or higher*
		Jan	37	*"@ row,column SAY ERASE" only clears one line in CP/M-80 formats*
	DEBUG	Jan	37	*Patches to access line 24 with @...SAY in versions 2.4 and higher*
	NUMERIC VARIABLES	Mar	40	*How to left-justify numeric variables*
	SUBDIRECTORIES	Mar	39	*Using logical drive designators as subdirectories with DOS 3.1*
dBASE III	ASSIST	Mar	40	*Commands executed in ASSIST can be recorded in HISTORY*
	COPY	Jan	38	*Method to COPY fields from one database file to another*
	DELETE	Jan	39	*Deleting records based on values in a related database file*
	SET PATH TO	Jan	40	*If SAFETY is ON, some commands will search the path before writing*
		Jan	40	*Commands which use the path to locate a file*
	SET SAFETY ON/OFF	Jan	40	*If ON, some commands will search the path before writing to disk*

Usage Tips

Product	Entry	Issue	Page	Description
dBASE III PLUS	@...SAY	Apr	42	*Control codes are counted as characters but not sent to the printer*
	ACCESS DISK	Apr	42	*Disk manufacturing error affected products with certain serial numbers*
	ASSIST	Apr	43	*Building search conditions with date fields*
		Apr	43	*Cannot build a search condition for blank spaces*
	CASE	Apr	43	*Omission of DO CASE or ENDCASE in a CASE structure affects commands*
	CONFIG.DB	Apr	44	*Using the keyword COMMAND in CONFIG.DB will alter the initial sign on*
	CONTROL CODES	Apr	42	*Control codes to the printer with @...SAYs are counted as character*
	CTOD()	Apr	43	*Non-date entries are accepted as arguments in ASSIST*
	DATE FIELDS	Apr	43	*Building search conditions with date fields in ASSIST*
	DO WHILE	Apr	44	*DO WHILE EOF()=.T. returns "Invalid Operator"*
	DOT PROMPT	Apr	44	*The command line now scrolls, eliminating the need for a semi-colon*
	ERROR MESSAGE	Apr	44	*"Invalid Operator": Results from improper syntax of DO WHILE statement*
	HISTORY	Mar	40	*Commands executed in ASSIST can be recorded in HISTORY*
	INPUT	Apr	44	*INPUT does not accept a Return as valid input*
	INTERACTIVE MODE	Apr	44	*Cannot continue dot prompt commands with a semi-colon*
	MODIFY VIEW	Apr	46	*Will return error message if any components of the view have changed*
	PRINTING	Apr	42	*Control codes to the printer with @...SAYs are counted as characters*
	SET DATE FRENCH	Apr	45	*This command is now the same as SET DATE BRITISH*
	SET FIELDS TO	Apr	45	*SET FIELDS TO <field list> automatically SETs FIELDS ON*
	SET ORDER TO	Apr	45	*Must reposition the record pointer after this command*
	SET SCOREBOARD	Apr	45	*Clears the entire screen*
	SET STATUS	Apr	45	*Clears the entire screen*
	SET VIEW TO	Apr	46	*Cannot make changes to any component parts of a view*
Developer's Release	DBRUN.COM	Jan	40	*This file, not DBRUN.EXE, should be renamed as per the documentation*
	DBRUN.EXE	Jan	40	*This hidden file will not show up with DIR and cannot be renamed*
	READ	Jan	40	*In Developer's Release, a READ must have active GETs*
		Jan	40	*Cannot issue a READ from within HISTORY in Developer's Release*
	REINDEX	Jan	40	*Must reissue the SET ORDER TO command if you REINDEX*
	SET ORDER TO	Jan	40	*Use REINDEX to SET ORDER TO its original state*

Usage Tips

Product	Entry	Issue	Page	Description
Framework II	.PAR FILES	Mar	43	*Converting .PAR files to .TCM files*
	.TCM FILES	Mar	43	*Converting .PAR files to .TCM files*
	@sum	Mar	42	*Cannot remove the last column/row referenced by @sum*
	@time	Jan	42	*Displaying times with the Numbers pull-down menu*
	APPS MENU	Mar	43	*Some Print Output Options have no effect on Mailmerge or Print Labels*
	DICTIONARY	Mar	42	*May take a long time to save the personal dictionary the first time*
	DOW JONES	Mar	42	*Copying fractions from text into spreadsheets*
	EXTENDED MEMORY	Mar	42	*Memory requirements for extended memory*
	FRACTIONS	Mar	42	*Copying fractions from text into spreadsheets*
	IBM 3270t	Jan	44	*This screen driver is found under "All Other IBM Drivers" in SETUP*
	IMPORT	Mar	41	*Importing Framework 1.x files to Framework II*
	LABELS	Mar	43	*Some Print Output Options have no effect on Mailmerge or Print Labels*
	MAILMERGE	Mar	43	*Some Print Output Options have no effect on Mailmerge or Print Labels*
	PAR FILES	Jan	44	*How to read .PAR files from Framework into Framework II*
	PRINTING	Mar	41	*How to print subscripts and superscripts from Framework II*
		Mar	43	*Some Print Output Options have no effect on Mailmerge or Print Labels*
	SETUP	Jan	42	*Selecting screen color parameters in SETUP*
	SIZING COLUMNS	Mar	42	*Changing several spreadsheet or database columns at one time*
	SPREADSHEETS	Mar	42	*Copying fractions from text into spreadsheets*
	SUBSCRIPTS	Mar	41	*How to print subscripts and superscripts from Framework II*
	SUPERSCRIPTS	Mar	41	*How to print subscripts and superscripts from Framework II*
	TELECOMMUNICATIONS	Mar	43	*Converting .PAR files to .TCM files*
	TEXT	Mar	42	*Copying fractions from text into spreadsheets*
	VIRTUAL MEMORY	Mar	41	*Cannot use virtual memory to import Framework 1.x files*

Anomalies

Product	Entry	Issue	Page	Description
dBASE II	APPEND FROM	Jan	45	*Cautions regarding APPENDing numeric data in version 2.43**
	COUNT	Feb	43	*In DOS ver. 3.x, COUNT may take longer to execute the second time*
	ERROR MESSAGE	Feb	42	*" 'TO' PHRASE NOT FOUND": Memory variable corruption*
	MEMORY VARIABLES	Feb	42	*Corruption of numeric memory variables in dBASE II, version 2.43**
	RENAME.COM	Feb	43	*This file is not on the RunTime disk, as the manual states*
	RUNTIME	Feb	43	*RENAME.COM is not on the RunTime disk, as the manual states*
	SKIP	Apr	47	*Cannot have a space preceding the argument of the SKIP function*
dBASE III	BROWSE	Jan	45	*Problems using the SEEK option with SET DELETED ON*
	DBRUN.OVL	Jan	46	*This file may not get copied during Developer's Release INSTALL*
	ERROR MESSAGE	Jan	46	*"Record is out of range": trying to GOTO a very large number*
	FIND	Jan	45	*With numeric INDEX, FIND <cstring> will FIND first zero valued record*
	GOTO	Jan	46	*Attempting to GOTO a very large number may lock the computer*
	INDEX	Jan	45	*With numeric INDEX, FIND <cstring> will FIND first zero valued record*
	INSTALL	Jan	46	*Developer's Release: INSTALL may not copy DBRUN.OVL*
	PARAMETERS	Jan	46	*Cannot wrap PARAMETERS line with semi-colon*
	RUNTIME+	Jan	46	*DBRUN.OVL may not get copied during Developer's Release INSTALL*
	SEEK	Jan	45	*Problems using the SEEK option in BROWSE with SET DELETED ON*
	SET DELETED ON	Jan	45	*Problems using the SEEK option in BROWSE with SET DELETED ON*
dBASE III PLUS	@...SAY/GET	Apr	48	*Should not use line 0 with SET SCOREBOARD ON*
	CREATE/MODIFY SCREEN	Apr	48	*The <new field> option always creates a one-character field*
	CREATE/MODIFY VIEW	Apr	48	*Cannot permanently delete a FILTER in MODIFY VIEW*
		Apr	48	*Can only include files from the current subdirectory*
	ERROR MESSAGE	Apr	48	*"Alias already in use": Accessing another subdirectory with views*
		Apr	48	*"Record not in index": Using SET CENTURY ON with DTOC() to index file*
	FOUND()	Apr	48	*Will return .F. if first record is deleted and SET DELETED is on*
	INDEX	Apr	48	*Cannot use SET CENTURY ON with DTOC() to create an index file*
	READKEY()	Apr	49	*Returns 270 if operation is terminated by Ctrl-End or Ctrl-W*
	SET SCOREBOARD	Apr	47	*Should not address line 0 with SET SCOREBOARD ON*

Letters

Product	Entry	Issue	Page	Description
dBASE II	CHR()	Apr	56	*Sending line feeds (without carriage returns) to the printer*
	DATA DISPLAY	Feb	44	*Old data displaying in a file with zero records*
	NUMERIC ACCURACY	Jan	49	*Versions prior to 2.43* have 10 digits of numeric accuracy*
	PRINTING	Apr	56	*Sending line feeds (without carriage returns) to the printer*
	REPORT FORM	Jan	48	*Incorrect totals in versions prior to 2.43**
	SED	Mar	44	*Work-around for hard tabs created in SED*
dBASE III	BROWSE	Jan	49	*Problems using the SEEK option with SET DELETED ON*
	CHARACTER STRINGS	Mar	45	*Removing preceding blanks from character strings in dBASE III*
	ERROR MESSAGE	Jan	48	*"NO FIND": Character index key length must be a multiple of 4*
		Jan	48	*"Internal error, illegal opcode ": Char INDEX must be multiple of 4*
	IMPORT/EXPORT	Feb	44	*Transferring data from Lotus 1-2-3 to dBASE III*
	INDEX	Apr	56	*Creating index files in a subdirectory*
		Jan	47	*Character INDEX key length must be a multiple of 4*
	INSTALL	Jan	48	*Problems with Tallgrass: set Duplicate Directories OFF*
	LABEL FORM	Mar	45	*Using two database files with LABEL FORM*
	LOTUS	Feb	44	*Transferring data from Lotus 1-2-3 to dBASE III*
	SED	Mar	44	*Work-around for hard tabs created in SED*
	SEEK	Jan	49	*Problems using the SEEK option in BROWSE with SET DELETED ON*
	SET	Jan	49	*Accessing SET parameters from within program control*
	SET DELETED ON	Jan	49	*Problems using the SEEK option in BROWSE with SET DELETED ON*
	SET PATH TO	Apr	56	*Cannot use SET PATH TO to create index files in subdirectories*
	SUBDIRECTORIES	Apr	56	*Creating index files in a subdirectory*
	TALLGRASS	Jan	48	*Problems INSTALLing: set duplicate directories OFF*
dBASE III PLUS	IIF()	Apr	57	*Using IIF() for conditional indexing*
	LTRIM()	Mar	45	*This function removes preceding blanks from character strings*
Framework II	ASSEMBLY LANGUAGE	Mar	46	*Using Framework II word processor to write assembly language programs*
	IMPORT	Apr	55	*How to import Lotus 1-2-3 Release 2 files*
	LOTUS 1-2-3	Apr	55	*How to import Lotus 1-2-3 Release 2 files*
	MEMORY-RESIDENT	Feb	45	*May have problems using memory-resident programs with Framework*
	TEXT FRAME	Mar	46	*A containing frame can hold more than one 64K text frame*

Sample Programs

Product	Entry	Issue	Page	Description
dBASE II	CSEED2.PRG	Jan	17	*Program to create a seed value for random number generation-dBASE II*
	EVAL2.PRG	Jan	22	*Calculates the standard deviation and mean value of random numbers-D2*
	HEC-DEC.CMD/PRG	Feb	18	*Converts hex program into a dBASE II POKE sequence*
	LEADZERO.PRG	Feb	20	*Program to replace leading blanks with zeros*
	MODSTUC.PRG	Jan	10	*Program to emulate MODIFY STRUCTURE*
	RANDOM2.PRG	Jan	18	*Program to generate a random number*
	WINNERS.PRG	Jan	20	*Program to select random winners from a database file*
dBASE III	CSEED3.PRG	Jan	19	*Program to create a seed value for random number generation*
	DECRYPT.PRG	Feb	33	*Data decryption program for dBASE III*
	ENCRYPT.PRG	Feb	31	*Data encryption program for dBASE III*
	EVAL3.PRG	Jan	23	*Calculates the standard deviation and mean value of random numbers-D3*
	FILEPROC.PRG	Mar	22	*Procedure file prints program listings*
	PRGPRINT.PRG	Mar	21	*Program to display menu selection for print program listings*
	RANDOM3.PRG	Jan	20	*Program to generate a random number*
dBASE III PLUS	COLOR.PRG	Mar	12	*Program to SET COLOR from a menu in dBASE III PLUS*
	DATABASE.PRG	Apr	28	*How to maintain database files in the Catalog management application*
	DISPLAY.PRG	Apr	30	*Program to manipulate data in the Catalog*
	ERR.PRG	Apr	32	*Program to handle errors in the Catalog management application*
	FREEARRAY.PRC	Apr	16	*Procedure to remove an array from memory*
	LISTFORM.PRG	Apr	23	*Program to list a database file in a vertical form*
	LISTHIST.PRG	Feb	8	*Program to LIST HISTORY to a file in dBASE III PLUS*
	LOADARRAY.PRC	Apr	14	*Procedure to load an array from a database file*
	MAIN.PRG	Apr	27	*Main driver program for the Catalog management application*
	MAKEARRAY.PRC	Apr	12	*Procedure to create a new array*
	NEWPAGE.PRG	Apr	25	*Program to print page top called from Listform.PRG*
	OPTION.PRG	Apr	29	*Program to allow access to index files and queries in the Catalog*
	SEEKARRAY.PRC	Apr	15	*Procedure to find an element within a defined array*
	VIEW.PRG	Feb	12	*Program to toggle between table view and form view*
	WRAP.PRG	Apr	24	*Program to wrap long strings, called from Listform.PRG*

Sample Programs

Product	Entry	Issue	Page	Description
Framework II	ALT-B (BOX MACRO)	Apr	39	*Macro to draw boxes in word frames*
	DATA TABLE	Feb	35	*Program to create data tables in Framework II*
	HEX	Apr	37	*FRED function to convert a decimal number to hexidecimal*
	LABELPGM	Mar	14	*Program to automate mailing labels*
	LSTRING	Jan	25	*Function to extract the left portion of a character string*
	MID STRING	Jan	31	*Macro to extract the mid portion of a string in word frames*
	PFS TRANSFER	Mar	37	*Program to transfer files from pfs:file to Framework II*
	RSTRING	Jan	26	*Function to extract the right portion of a character string*
	RUNDBASE	Mar	27	*Running dBASE from within Framework II*
	STRING FUNCTIONS	Jan	24	*User-defined string functions used in spreadsheet and database frames*
	TABSTOP	Mar	18	*Macro for putting variable tab stops into word frames*
	TABSTRIP	Jan	33	*Program to remove tabs from a word frame*
Other	CHOOSE.COM	Mar	33	*An assembly language routine used to choose menu options with MENU.BAT*
	MENU.BAT	Mar	32	*DOS batch file is controlling program for front-end menu system*
	MENU.DAT	Mar	33	*DOS batch file, called from MENU.BAT, displays menu screen*

Miscellaneous

Subject	Issue	Page	Description
CORRIGENDA	Apr	19	Corrections to the December 1985 issue
	Jan	36	Corrections to September 1985, November 1985, and December 1985 issues
	Feb	27	Corrections to the March 1985 issue
	Mar	3	Corrections to the July 1985, October 1985, and December 1985 issues
MULTIMATE	Apr	50	Features of MultiMate Advantage update (version 3.60)
	Apr	52	Where to look when you have technical difficulties
	Apr	52	About MultiMate Technical Support—Supported products, when to call
	Apr	53	Customer Service versus Technical Support
ON-LINE SUPPORT	Jan	3	Information pertaining to CompuServe, The Source, and Onepoint
SUBSIDIARY OFFICES	Jan	56	List of Ashton-Tate international subsidiary offices
TRAINING CENTERS	Jan	54	Listing of Ashton-Tate Authorized Training Centers
	Mar	47	Listing of Ashton-Tate Authorized Training Centers

Notes

Notes

Notes

TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

ADDRESS CORRECTION REQUESTED
RETURN POSTAGE GUARANTEED

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 27
CULVER CITY, CA